Summer 8-2013

Data Collection and Capacity Analysis in Large-scale Wireless Sensor Networks

Shouling Ji
Georgia State University

DATA COLLECTION AND CAPACITY ANALYSIS IN LARGE-SCALE WIRELESS SENSOR NETWORKS

by

SHOULING JI

Under the Direction of Dr. Yingshu Li

ABSTRACT

In this dissertation, we study data collection and its achievable network capacity in Wireless Sensor Networks (WSNs). Firstly, we investigate the data collection issue in dual-radio multi-channel WSNs under the protocol interference model. We propose a multi-path scheduling algorithm for snapshot data collection, which has a tighter capacity bound than the existing best result, and a novel continuous data collection algorithm with comprehensive capacity analysis. Secondly, considering most existing works for the capacity issue are based

on the ideal deterministic network model, we study the data collection problem for practical probabilistic WSNs. We design a cell-based path scheduling algorithm and a zone-based pipeline scheduling algorithm for snapshot and continuous data collection in probabilistic WSNs, respectively. By analysis, we show that the proposed algorithms have competitive capacity performance compared with existing works. Thirdly, most of the existing works studying the data collection capacity issue are for centralized synchronous WSNs. However, wireless networks are more likely to be distributed asynchronous systems. Therefore, we investigate the achievable data collection capacity of realistic distributed asynchronous WSNs and propose a data collection algorithm with fairness consideration. Theoretical analysis of the proposed algorithm shows that its achievable network capacity is order-optimal as centralized and synchronized algorithms do and independent of network size. Finally, for completeness, we study the data aggregation issue for realistic probabilistic WSNs. We propose order-optimal scheduling algorithms for snapshot and continuous data aggregation under the physical interference model.

INDEX WORDS: Wireless sensor networks, Data collection, Data aggregation, Delay analysis, Capacity analysis, Protocol interference model, Physical interference model, Generalized physical interference model, Deterministic network model, Probabilistic network model

DATA COLLECTION AND CAPACITY ANALYSIS IN LARGE-SCALE WIRELESS SENSOR NETWORKS

by

SHOULING JI

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

in the College of Arts and Sciences

Georgia State University

2013

DATA COLLECTION AND CAPACITY ANALYSIS IN LARGE-SCALE WIRELESS SENSOR NETWORKS

by

SHOULING JI

Committee Chair: Dr. Yingshu Li

Committee: Dr. Raheem Beyah

Dr. Xiaojun Cao

Dr. Yi Zhao

Electronic Version Approved:

Office of Graduate Studies

College of Arts and Sciences

Georgia State University

August 2013

DEDICATION

This dissertation is dedicated to my parents and my grandparents.

ACKNOWLEDGEMENTS

I would like to show my great gratitude to all of those people who supported and helped me to complete my dissertation. Their generous help made this dissertation possible.

First of all, I am deeply grateful to my advisor, Dr. Yingshu Li, for her inspiration, guidance, thoughts, and friendship. Dr. Li always gives me the greatest tolerance and support during my graduate studies. The discussion with her is thought-provoking and the source of my ideas and inspirations.

I would like to thank my committee members, Dr. Raheem Beyah, Dr. Xiaojun Cao, and Dr. Yi Zhao. Dr. Beyah provides me many valuable suggestions on my research direction selection and also shares his great ideas on my research. Dr. Cao teaches me a lot on how to find research issues and how to resolve them. Dr. Zhao teaches me many useful mathematical theories and methods. Dr. Cao and Dr. Zhao gave me a lot of help and support in their classes as well.

Many thanks go to Dr. Zhipeng Cai. By his broad academic vision and incisive academic perspectives, Dr. Cai provides me many suggestions on my research with foresight and sagacity. Besides research, Dr. Cai also gives me a lot of help on my study and career planning.

I also would like to thank the professors and staffs at our department, especially Dr. Yi Pan, for his profound insights, Dr. Raj Sunderraman, for his help on my Ph.D. study, teaching, and research, Dr. Anu Bourgeois, for her suggestions on writing papers, and Ms. Tammie Dudley and Mr. Shaochieh Ou, for their patient help.

I would like to express my appreciation to Prof. Jianzhong Li and Prof. Jinbao Li, who brought me to the research community and encouraged me to make progress in the research world.

Special thanks go to my group members, fellow students, and friends at both Georgia State University and the Communications Assurance & Performance (CAP) Group in Geor-

gia Institute of Technology, Dr. Chunyu Ai, Dr. Jing (Selena) He, Dr. Selcuk Uluagac, Yueming Duan, Meng Han, Mingyuan Yan, Guoliang Liu, Xiaojing Liao, Sang Shin Jung, Wenyi Liu, Troy Nunnally, Ramalingam Chandrasekar, Sakthi Radhakrishnan, Venkatachalam Subramanian, Supreeth Sathyanarayana, Yunmei Lu, Xuhong Zhang, Chenguang Kong, and Nana Li, who provided me invaluable suggestions and help besides on research. Thank all of my friends.

Last but not least, I would like to thank my parents, my grandparents, and my family for their continuous support, understanding, and help. They are there whenever I need them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

- WSNs - Wireless Sensor Networks
- CDG - Compressive Data Gathering
- BFS - Breadth-First-Search
- DS - Dominating Set
- CDS - Connected Dominating Set
- MPS - Multi-Path Scheduling
- PS - Pipeline Scheduling
- i.i.d. - independent and identically distributed
- SINR - Signal-to-Interference-plus-Noise Ratio
- CPS - Cell-based Path Scheduling
- ZPS - Zone-based Pipeline Scheduling
- CTCS - Compatible Transmission Cell Set
- SIR - Signal-to-Interference Ratio
- DDC - Distributed Data Collection
- CR - Carrier-sensing Range
- DDA - Distributed Data Aggregation
- PCR - Proper Carrier-sensing Range
- CAS - Cell-based Aggregation Algorithm
- LAS - Level-based Aggregation Algorithm

PART 1

INTRODUCTION

1.1 Background

Recently, the developments of embedded computing technology, distributed information processing technology, wireless communication technology, and Micro-Electro-Mechanical Systems (MEMS) enable the emerging of low-cost, low-power, multi-functional wireless sensor nodes, which have the computing, communication, and sensing capabilities [2]. Wireless Sensor Networks (WSNs) consist of spatially distributed autonomous wireless sensor nodes to monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion, or/and pollutants, and to cooperatively pass their data through the network to a base station (sink) [3]. Ever since WSNs emerged, plenty of applications are developed based on them, e.g. area monitoring, air pollution monitoring, forest fires detection, greenhouse monitoring, landslide detection, machine health monitoring, data logging, water/wasterwater monitoring, structural monitoring, etc. [2][3]. Due to their wide applications, WSNs attract extensive interests from both the research communities and the industry.

One of the most important services provided by WSNs is to gather data from the physical world. Therefore, in this work, we focus on designing and implementing data collection algorithms and analyzing the achievable data collection capacity of the proposed algorithms.

1.2 Characteristics of WSNs

Wireless sensor nodes, the basic building block of WSNs, are usually composed of six parts: power unit, sensing unit, processing unit, storage, communication unit, and software. However, the computing, communication, and storage capabilities of a sensor node are very

limited, although it can be viewed as a small computer. Therefore, different from traditional wireless mesh and ad hoc networks, WSNs have many distinguished characteristics, which bring many challenge issues to the research community and industry. Specifically, some typical characteristics of WSNs can be summarized as follows.

1. **Large-Scale Wireless Networks.** A WSN may consist of thousands of sensor nodes and the deploy region of a WSN may be very large. Consequently, it is a challenge work to maintain such a huge network. When design algorithms for WSNs, the robustness and dynamical scalability of the algorithms should be considered.

2. **Limited Energy.** Wireless sensor nodes are usually battery-powered, and thus the available energy for a node is very limited. Furthermore, WSNs are large-scale networks and the deploy regions of WSNs are ever-changing, sometimes even the hazardous places that human intervention is not desirable or feasible. Therefore, the batteries of sensor nodes are not replaceable. Hence, how to use the limited precious energy of wireless sensor nodes is one of the most important concern when design algorithms for WSNs.

3. **Limited Communication Capability.** The transmission range of a sensor nodes is varied from tens of meters to hundreds of meters, which is highly depend on the geographical environments and the natural causes. The bandwidth of a sensor node is also very limited. Consequently, how to finish the expected tasks under the constraint of limited communication capability is a challenge issue in WSNs.

4. **Limited Computing and Storage Capabilities.** The computing, processing, and storage capabilities of sensor nodes are very limited. Thus, only some basic data processing and computing tasks can be finished on a node. Meanwhile, the memory and storage space of sensor nodes are also very limited, where some temporary data can be stored. Therefore, how to effectively finish some complicated tasks and cooperatively store large-scale of data is a research issue in WSNs.

5. **Dynamic Network.** As mentioned before, WSNs are large-scale networks. During the working process of a WSN, some nodes may die due to exhaust their energy or damaged by some other causes, and some new nodes may come to join the network. Hence, how to deal with this dynamics for WSNs and make the network adapt the changes is a challenge issue when design algorithms and protocols for WSNs.

6. **Huge Data Flows.** The data produced by the sensor nodes by viewed as data flows. Intuitively, as time goes on, huge data flows are generated by a WSN. Among these data flows, there may be a lot of redundant data. Considering the limitations of sensors nodes on computing, communication, and storage capabilities, how to manage, query, analyze, and utilize these data is another challenge works for researchers.

In summary, the characteristics of WSNs make solutions for traditional wireless networks unsuitable for WSNs. They also introduce many challenge issues for researchers.

1.3 Research Progress on Data Gathering and Capacity Analysis in Wireless Networks

Generally speaking, data gathering in wireless networks can be categorized as *data collection* [1][4][5][6], which gathers all the data from a network without any data aggregation or merging, and *data aggregation* [7][8][9][10][11], which obtains some aggregation values, e.g. MAX, MIN, SUM, etc. To evaluate network performance, *network capacity*, which reflects the data transmission/collection/aggregation/broadcast rate, is usually employed, e.g. multicast capacity [12][13][14], unitcast capacity [15][16][17], broadcast capacity [18], data collection capacity [1][4][5], etc. In this section, we summarize the related works and advances on data gathering and capacity analysis in wireless networks.

1.3.1 Capacity for Single-Radio Single-Channel Wireless Networks

Following the seminal work [19] by Gupta and Kumar, extensive works emerged to study the network capacity issue. The works in [20]-[21] focus more on the MAC layer to

improve the network capacity. In [20], the network capacity with random-access scheduling is investigated. In this work, each link is assigned a channel access probability. Based on which some simple and distributed channel access strategies are proposed. Another similar work is [17], in which the authors studied the capacity of CSMA wireless networks. The authors formulated the models of a series of CSMA protocols and study the capacity of CSMA scheduling versus TDMA scheduling. They also proposed a CSMA scheme which combines a backbone-peripheral routing scheme and a dual carrier-sensing and dual channel scheme. In [22], the authors considered the scheduling problem where all the communication requests are single-hop and all the nodes transmit at a fixed power level. They proposed an algorithm to maximize the number of links in one time-slot. Unlike [22], the authors in [21] considered the power-control problem. A family of approximation algorithms were presented to maximize the capacity of an arbitrary wireless networks.

The works in [12], [13], [14], [15], [16], and [23] study the multicast and/or unicast capacity of wireless networks. The multicast capacity for wireless ad hoc networks under the protocol interference model and the Gaussian channel model are investigated in [12] and [13] respectively. In [12], the authors showed that the network multicast capacity is $\Theta(\sqrt{\frac{n}{\log n}} \cdot \frac{W}{k})$ when $k = O(\frac{n}{\log n})$ and is $\Theta(W)$ when $k = \Omega(\frac{n}{\log n})$, where W is the bandwidth of a wireless channel, n is the number of the nodes in a network, and k is the number of the nodes involved in one multicast session. In [13], the authors showed that when $k \leq \theta_1 \frac{n}{(\log n)^{2\alpha+6}}$ and $n_s \geq \theta_2 n^{1/2+\beta}$, the capacity that each multicast session can achieve is at least $c_8 \frac{\sqrt{n}}{n_s \sqrt{k}}$, where k is the number of the receivers in one multicast session, n is the number of the nodes in the network, n_s is the number of the multicast sessions, θ_1, θ_2 and c_8 are constants and β is any positive real number. Another similar work [14] studies the upper and lower bounds of multicast capacity for hybrid wireless networks consisting of ordinary wireless nodes and multiple base stations connected by a high-bandwidth wired network. Considering the problem of characterizing the unicast capacity scaling in arbitrary wireless networks, the authors proposed a general cooperative communication scheme in [15]. The authors also presented a family of schemes that address the issues between multi-hop and cooperative

communication when the path-loss exponent is greater than 3. In [16], the authors studied the balanced unicast and multicast capacity of a wireless network consisting of n randomly placed nodes, and obtained the characterization of the scaling of the n^2-dimensional balanced unicast and $n2^n$-dimensional balanced multicast capacity regions under the Gaussian fading channel model. A more general (n, m, k)-casting capacity problem was investigated in [23], where n, m and k denote the total number of the nodes in the network, the number of destinations for each communication group, and the actual number of communication-group members that receive information respectively. In [23], the upper and lower bounds for the (n, m, k)-cast capacity were obtained for random wireless networks.

In [24], the authors investigated the network capacity scaling in mobile wireless ad hoc networks under the protocol interference model with infrastructure support. In [25], the authors studied the network capacity of hybrid wireless networks with directional antenna and delay constraints. Unlike previous works, the authors in [26] studied the capacity of multi-unicast for wireless networks from the algorithmic aspects, and they designed provably good algorithms for arbitrary instances. The broadcast capacity of wireless networks under the protocol interference model is investigated in [27], where the authors derived the upper and lower bounds of the broadcast capacity in arbitrary connected networks. When the authors in [28] studied the data gathering capacity of wireless networks under the protocol interference model, they concerned the per source node throughput in a network where a subset of nodes send data to some designated destinations while other nodes serve as relays. To gather data from WSNs, a multi-query processing technology is proposed in [29]. In that work, the authors considered how to obtain data efficiently with data aggregation and query scheduling. Under different communication organizations, the authors in [30] derived the many-to-one capacity bound under the protocol interference model. Another work studied the many-to-one capacity issue for WSNs is [31], where the authors considered to use data compression to improve the data gathering efficiency. They also studied the relation between a data compression scheme and the data gathering quality. In [32], the authors studied the scaling laws of WSNs based on an antenna sharing idea. In that work, the authors derived the

many-to-one capacity bounds under different power constraints. In [33], the authors studied the multicast capacity of MANETs under the physical interference model, called motioncast. They considered the network capacity of MANETs in two particular situations, which are the LSRM (local-based speed-restricted) model and the GSRM (global-based speed-restricted) model. The multi-unicast capacity of wireless networks is studied in [34] via percolation theory. By applying percolation theory, the authors obtained a tighter capacity bound for arbitrary wireless networks.

The data collection capacity of WSNs is studied in [35], [4], [1], [36], [37], [38], [39], [40], [41], [42], [43], [44], [45], etc. In [35], the authors considered the collision-free delay-efficient data gathering problem. Furthermore, they proposed a family of path scheduling algorithms to collect all the data to the sink and obtained the network capacity through theoretical analysis. The authors of [4] extended the work of [35]. They derived tighter upper and lower bounds of the capacity of data collection for arbitrary WSNs. [1] is a work studying how to distribute the data collection task to the entire network to achieve load balancing. In this work, all the sensors transmit the same number of data packets during the data collection process. In [36] and [37][38][39][40], the authors investigated the capacity of data collection for WSNs under protocol interference model and physical interference model, respectively. They proposed a grid partition method which divides the network into small grids to collect data and then derived the network capacity. In [42], the authors studied the distributed data collection problem in asynchronous wireless networks. They proposed a distributed data collection algorithm and theoretically analyzed the delay and capacity performance of the proposed algorithm, which are proven to be order-optimal. The worst-case capacity of data collection of a WSN is studied in [41] under the physical and protocol interference models. In [43], [44] and [45], the data gathering issue for cognitive radio networks is investigated and analyzed.

The capacity and energy efficiency of wireless ad hoc networks with multi-packet reception under the physical interference model is investigated in [46]. With the multi-packet reception scheme, a tight bound of the network capacity is obtained. Furthermore, the au-

thors showed that a tradeoff can be made between increasing the transport capacity and decreasing the energy efficiency. In [47], a scheduling partition method for large-scale wireless networks is proposed. This method decomposes a large network into many small zones, and then localized scheduling algorithms which can achieve the order optimal capacity as a global scheduling strategy are executed in each zone independently. A general framework to characterize the capacity of wireless ad hoc networks with arbitrary mobility patterns is studied in [48]. By relaxing the "homogeneous mixing" assumption in most existing works, the capacity of a heterogeneous network is analyzed. Another work [49] studies the relationship between the capacity and the delay of mobile wireless ad hoc networks, where the authors studied how much delay must be tolerated under a certain mobile pattern to achieve an improvement of the network capacity.

1.3.2 Capacity for Multi-Radio Multi-Channel Wireless Networks

Since wireless nodes can be equipped with multiple radios, and each radio can work over multiple orthogonal channels, multi-radio multi-channel wireless networks attract many research interests recently [50][51][52][53]. In [50], the authors studied the data aggregation issue in multi-channel WSNs under the protocol interference model. Particularly, they designed a constant factor approximation scheme for data aggregation in multi-channel WSNs modeled by Unit Disk Graphs (UDGs). Unlike [50], we study the data collection capacity issue for WSNs. In [51], [52], and [53] the authors investigated the joint channel assignment and routing problem for multi-radio wireless mesh networks, software-defined radio networks, and multi-channel ad hoc wireless networks, respectively. They focused on the channel assignment and routing issues, while in data collection, especially continuous data collection, we focus on how to solve the data accumulation problem at the sensors near the sink to improve the achievable network capacity.

The issue of the capacity of multi-channel wireless networks also attracts a lot of attention [54][55][56][57][58]. In [54] and [55], the authors studied the connectivity and capacity problem of multi-channel wireless networks. They considered a multi-channel wireless net-

work under constraints on channel switching, proposed some routing and channel assignment strategies for multiple unicast communications and derived the per-flow capacity. The multicast capacity of multi-channel wireless networks is studied in [56]. In this work, the authors represented the upper bound capacity of per multicast as a function of the number of the sources, the number of the destinations per multicast, the number of the interfaces per node, and the number of the available channels. Subsequently, an order-optimal scheduling method is proposed under certain circumstances. In [57], the authors first proposed a multi-channel network architecture, called MC-MDA, where each node is equipped with multiple directional antennas, and then obtained the capacity of multiple unicast communications under arbitrary and random network models. The impact of the number of the channels, the number of the interfaces and the interface switching delay on the capacity of multi-channel wireless networks is investigated in [58]. In this work, the authors derived the network capacity under different situations for arbitrary and random networks.

In [5][59], we studied the snapshot and continuous data collection issues and their achievable capacities for dual-radio multi-channel WSNs under the protocol interference model. First, we proposed a novel multi-path scheduling algorithm for snapshot data collection. By theoretical analysis, we showed our snapshot data collection algorithm has a better performance than the state-of-the-art method. Subsequently, we pipeline-based continuous data collection method, which also proved to have a good capacity performance.

1.3.3 Data Aggregation

Ever since the data aggregation problem is raised, extensive research has been conducted on this issue ([7], [8], [60], [61], [62], [63], [64], [65], and references therein), especially for the Minimum-Latency Aggregation Schedule (MLAS) problem, which tries to obtain a data aggregation schedule with the objective to minimize the latency (minimize M). In [60], [61] and [7], several centralized data aggregation algorithms are proposed under the Unit Disk Graph (UDG) model and the protocol interference model. Chen et al. [60] proved that the MLAS problem is NP-hard. Furthermore, they designed a $(\Delta - 1)$-approximation algorithm

for this problem, where Δ is maximum degree of the topological graph of the network. Subsequently, Huang et al. [61] proposed another data aggregation algorithm which has a better performance. By analysis, they showed that the delay of their algorithm is upper bounded by $23\mathfrak{R}+\Delta-18$ ($\mathfrak{R} \sim L$ and L is the height of the data aggregation tree), where $\mathfrak{R}$ is the network radius. Recently, Wan et al. [7] proposed three data aggregation algorithms of latency upper bounded by $15\mathfrak{R}+\Delta-4$, $2\mathfrak{R}+O(\log \mathfrak{R})+\Delta$, and $(1+O(\log \mathfrak{R}/\sqrt[3]{\mathfrak{R}}))\mathfrak{R}$, respectively. Xu et al. [62] studied periodic query scheduling for data aggregation with minimum delay consideration. They designed centralized aggregation scheduling algorithms under various wireless interference models, and analyzed the induced delay of each algorithm. As we have already known, centralized algorithms have many shortcomings in distributed wireless networks. To overcome these shortcomings, some state-of-the-art distributed algorithms are proposed under the UDG model and the protocol interference model [63][64][65]. In [63], Yu et al. proposed a distributed Connected Dominating Set (CDS)-based data aggregation schedule algorithm with latency upper bounded by $24\mathfrak{D}+6\Delta+16$, where $\mathfrak{D}$ is the network diameter. Xu et al. [64] also proposed a distributed data aggregation algorithm with a better latency bound of $16\mathfrak{R}'+6\Delta-14$, where $\mathfrak{R}'$ is the inferior network radius which satisfies $\mathfrak{R}' \leq \mathfrak{R} \leq \mathfrak{D} \leq 2\mathfrak{R}'$. The most recently published distributed data aggregation algorithm is [65], in which Li et al. proposed an aggregation scheme of latency upper bounded by $16\mathfrak{R}'+\Delta-14$.

1.3.4 Remarks

Unlike the existing works, in this dissertation, we study the data collection and data aggregation issues and their achievable capacities for WSNs under different scenarios, and proposed a series of data collection and aggregation algorithms. We theoretically analyze all the proposed algorithms, and obtain their delay and capacity performance. We also conduct extensive simulations to validate the performance of all the proposed algorithms, and compare them with the state-of-the-art methods.

Particularly, some or all the following aspects distinguish the works in this dissertation

from existing literatures.

1. Most of the above mentioned works are specifically for single-radio single-channel wireless networks, while our work (Part 2) considers the network capacity for dual-radio multi-channel WSNs.

2. Our work (Part 2, Part 3) is the dedicated one that investigates the network capacity for continuous data collection in detail under the protocol interference model/physical interference model, whereas most of the previous works study the network capacity for multicast or/and unicast, etc, which are different communication modes from the snapshot data collection, especially the continuous data collection. For the works that study the data collection capacity of wireless networks, they focus on the snapshot data collection problem which is a special case of continuous data collection. Compared with them, the results proposed in this dissertation are more universal.

3. Most of the previous works considered the network capacity issues under the deterministic network model, which is not practical due to the existence of plenty of lossy links. Unlike them, we study the network capacity issue under the probabilistic network model (Part 3, Part 5), which is more realistic.

4. To the best of our knowledge, this work (Part 4) is the first attempt to address the distributed data collection problem with capacity analysis for asynchronous wireless sensor networks, which is more complicated, however, more practical. As summarized in Section 1.3, the existing works study the data collection capacity issue based on centralized and synchronized scheduling/algorithms. On the other hand, we propose a scalable and order optimal asynchronous distributed data collection algorithm in Part 4. This demonstrates that asynchronous distributed data collection schemes can also achieve order optimal data collection capacity as synchronized and centralized algorithms do.

5. For completeness, we also consider the data gathering issue with data aggregation, and

propose centralized and distributed data aggregation algorithms (Part 5), which are proven to be order-optimal.

1.4 Organization

The rest of this dissertation is organized as follows: Part 2 studies the snapshot and continuous data collection issues for dual-radio multi-channel WSNs, where a multi-path scheduling algorithm for snapshot data collection and a pipeline-based scheduling algorithm for continuous data collection are proposed and analyzed. Part 3 investigates the data collection issue for practical probabilistic WSNs, where a snapshot data collection algorithm and a continuous data collection algorithm are proposed and analyzed under the physical interference model. Part 4 studies the distributed data collection problem for asynchronous WSNs. In that part, an asynchronous distributed data collection algorithm is proposed. By theoretical analysis, we show that the proposed distributed data collection algorithm can also surprisingly achieve order-optimal data collection capacity as centralized and synchronized algorithms do. For completeness, Part 5 studies the data aggregation issue for probabilistic WSNs, where two data aggregation algorithms are proposed and analyzed for snapshot and continuous data aggregation, respectively. Finally, the dissertation is concluded in Part 6.

PART 2

CONTINUOUS DATA COLLECTION AND CAPACITY IN DUAL-RADIO MULTI-CHANNEL WIRELESS SENSOR NETWORKS

2.1 Introduction

Wireless Sensor Networks (WSNs) are mainly used for collecting data from the physical world. Data gathering can be categorized as *data aggregation* [7]-[61], which obtains aggregated values from WSNs, e.g. maximum, minimum or/and average value of all the data, and *data collection* [1]-[5], which gathers all the data from a network without any data aggregation. For data collection, the union of all the sensing values from all the sensors at a particular time instance is called a *snapshot* [38][4][5]. The problem of collecting all the data of one snapshot is called *snapshot data collection*. Similarly, the problem of collecting multiple continuous snapshots is called *continuous data collection*. Different from wired networks, WSNs suffer from the interference problem, which degrades the network performance. Consequently, *network capacity*, which can reflect the achievable data transmission rate, is usually used as an important measurement to evaluate network performance. Particularly, for a data collection WSN, we use the average data receiving rate at the sink during the data collection process, referred to as *data collection capacity* [38][4][5], to measure its achievable network capacity, i.e. data collection capacity reflects how fast data been collected to the sink. In this part, we study the snapshot and continuous data collection problems, as well as their achievable capacities for WSNs.

After the first work [19], extensive works emerged to study the network capacity issue for variety of network scenarios, e.g. multicast capacity [12]-[14], unicast capacity [15], [16], broadcast capacity [18][66], snapshot data collection capacity [1]-[37], [4], etc. Most of the previous studies on network capacity are for single-radio single-channel WSNs [1]-[37], [4], [19]-[49], where a network consists of a number of nodes, each with only one radio, and all

the nodes communicate over a common single channel. Because of the inherent limitations of such networks, transmissions suffer from the *radio confliction* problem [67]-[58] and the *channel interference* problem [54]-[57], [58] seriously. This degrades network performance significantly. The radio confliction problem is caused by the fact that each node is equipped with only one radio, which means a node can only work on a *half-duplex* mode, i.e. this node cannot receive and transmit data simultaneously. The channel interference problem is caused by all the nodes working over a common channel. When one node transmits data, all the other nodes within its interference radius cannot receive any other data and all the other transmissions interfere with this transmission cannot be carried out simultaneously. Fortunately, many current off-the-shelf sensor nodes are capable of working over multiple orthogonal channels, e.g. IEEE 802.11 b/g standard supports 3 orthogonal channels and IEEE 802.11a standard supports 13 orthogonal channels [68], [57] respectively, which can greatly mitigate the channel interference problem. Furthermore, with the development of hardware technologies and the decreasing of hardware cost, a sensor node can be equipped with multiple radios. This helps with solving the radio confliction problem. Therefore, multi-radio multi-channel WSNs are currently becoming more and more attractive [67]-[58].

Different from the previous works which investigate the capacity issues for single-radio single-channel WSNs, we study the network capacity problem for both continuous data collection and snapshot data collection in dual-radio multi-channel WSNs under the *protocol interference model*. Similarly as [4], we define *capacity* as the data rate at the sink to continuously receive data from sensor nodes. We propose two channel scheduling algorithms for both continuous data collection and snapshot data collection, respectively, in this part. The motivation of this part lies in the fact that dual-radio multi-channel WSNs can make nodes work in a *full-duplex* manner without incurring high hardware cost, while the channel interference problem can be mitigated significantly. To the best of our knowledge, most of the previous works focus on addressing the snapshot data collection capacity problem, while this part is the dedicated one investigating the continuous data collection capacity problem in detail under the protocol interference model. Besides, this part is suitable for dual-radio

multi-channel WSNs. The main contributions of this part are as follows:

1. For the snapshot data collection problem in single-radio multi-channel WSNs, we propose a new *multi-path scheduling* algorithm. We prove that this algorithm can achieve the order-optimal network capacity $\Theta(W)$ and has a tighter lower bound $\frac{W}{2\lceil(1.81\rho^2+c_1\rho+c_2)/H\rceil}$ compared with the previously best result in [4], which is $\frac{W}{8\rho^2}$, where W is the channel bandwidth, H is the number of orthogonal channels, ρ is the ratio of the interference radius over the transmission radius of a node, $c_1 = \frac{2\pi}{\sqrt{3}} + \frac{\pi}{2} + 1$, and $c_2 = \frac{\pi}{\sqrt{3}} + \frac{\pi}{2} + 2$.

2. We propose a novel *pipeline scheduling* algorithm that combines *Compressive Data Gathering* (CDG) [1] and *pipeline* together, which significantly improves the continuous data collection capacity for dual-radio multi-channel WSNs. We also prove that the achievable asymptotic network capacity of this algorithm in a long-run is $\frac{nW}{12M\lceil(3.63\rho^2+c_3\rho+c_4)/H\rceil}$ when $\Delta_e \leq 12$ or $\frac{nW}{M\Delta_e\lceil(3.63\rho^2+c_3\rho+c_4)/H\rceil}$ when $\Delta_e > 12$, where n is the number of the sensors, M is a constant value and usually $M \ll n$, Δ_e is the maximum number of the leaf nodes having a same parent in the routing tree (i.e. data collection tree), $c_3 = \frac{8\pi}{\sqrt{3}}+\pi+2$, and $c_4 = \frac{8\pi}{\sqrt{3}}+2\pi+6$. A straightforward upper bound of data collection of a dual-radio WSN is $2W$, since a dual-radio sink can simultaneously receive two packets at most. Whereas, thanks to the benefit brought by the pipeline technique and CDG, analysis shows that our pipeline scheduling algorithm can even achieve a capacity higher than $2W$.

3. For completeness, we also examine the performance of the proposed pipeline scheduling algorithm in single-radio multi-channel WSNs, denoted by the *single-radio-based pipeline scheduling* algorithm. Theoretical analysis shows that for a long-run continuous data collection, the lower bound of the achievable asymptotic network capacity of the single-radio-based pipeline scheduling algorithm for single-radio multi-channel WSNs is $\frac{nW}{16M\lceil(3.63\rho^2+c_3\rho+c_4)/H\rceil}$ when $\Delta_e \leq 12$ or $\frac{nW}{M(\Delta_e+4)\lceil(3.63\rho^2+c_3\rho+c_4)/H\rceil}$ when $\Delta_e > 12$.

4. The simulation results indicate that the proposed algorithms have a better snapshot

data collection capacity compared with the previously best works. Particularly, when $\rho = 2$ and $H = 3$, for snapshot data collection in a WSN with 4000 nodes, the improvements of the capacity of our multi-path scheduling algorithm are 74.3% and 29% compared with BFS [4] and SLR [54] respectively. For continuous data collection in a WSN with 10000 nodes, our pipeline scheduling algorithm achieves a capacity 7.6 times of that of CDG [1], 22.8 times of that of BFS [4] and 19.4 times of that of SLR [54], respectively.

The rest of this part is organized as follows: Section 2.2 introduces the network model and preliminaries. The multi-channel scheduling algorithm for snapshot data collection in single-radio multi-channel WSNs is proposed and analyzed in Section 2.3. Section 2.4 presents a novel multi-channel scheduling algorithm for continuous data collection and its theoretical achievable asymptotic network capacity. The simulations to validate the performance of the proposed algorithms are shown in Section 2.5. We conclude this part and point out possible future research directions in Section 2.6.

2.2 Network Model and Preliminaries

In this section, we describe the network model and assumptions, construct the routing tree used for data collection, and introduce some necessary preliminaries. For the frequently used notations in this part, we list them in Table 2.1.

2.2.1 Network Model

We consider a WSN consisting of n sensors and one sink, represented by a connected undirected graph $G = (V, E)$, where V is the set of all the nodes in the network and E is the set of all the possible links among the nodes in V. Every sensor in the WSN produces one packet in a snapshot (defined in the subsequent paragraph). Each sensor has two radios and each radio has a fixed transmission radius normalized to one and a fixed interference radius, denoted by ρ, $\rho \geq 1$. Since we use the protocol interference model, for any receiving node v, v can receive a packet successfully from a transmitting node u if $\|u - v\| \leq 1$ and there is no

Table 2.1 Notations used in this part.

Notation	Description
$G(V, E)$	The network topology graph, V is the set of all the nodes, E is the set of all the possible links
n	The number of sensors in a WSN
ρ	The interference radius
$\lambda_1, \cdots, \lambda_H$	The H available orthogonal channels
W	The bandwidth of a channel
b	The size of a data packet
t	A time slot
τ	The time consumption of snapshot/continuous data collection
N	The number of snapshots in a continuous data collection
Υ	The snapshot/continuous data collection capacity
D/C	The set of dominators/connectors
G'	The graph constructed by nodes in D
L'	The radius of G'
T	The data collection tree
$\mathfrak{R}(A)$	The conflicting graph of A
$\Delta(\cdot)/\delta(\cdot)$	The maximum/minimum degree of a graph
Δ_e	The maximum number of leaf nodes having a same parent in T
$\delta^*(\cdot)$	The inductivity of a graph
β_r	The number of the dominators within a half-disk with radius r

other node s satisfying $\|s - v\| \leq \rho$ and trying to transmit a packet simultaneously over the same channel with u. Here $\|-\|$ is the Euclidean distance. Furthermore, we say two links are *interfering links* if at least one transmission over them will fail if they transmit data simultaneously. Each radio can work over H orthogonal channels, denoted by $\lambda_1, \lambda_2, \ldots, \lambda_H$ respectively. A fixed data-rate channel model [4] is adopted in this part, which means each sensor can transmit at a rate of W bits/second over a wireless channel. The size of all the packets transmitted in the network is set to be b bits. We also assume that all the transmissions are synchronized and the size of a time slot is $t = b/W$ seconds.

We formally define the problem follows. For a WSN consisting of n sensors and one sink, every sensor produces a data packet with b bits at a particular time instant. The union of all the n data packets produced by the n sensors at a particular time instant is called a *snapshot*. The process to collect all the data of a snapshot to the sink is called *snapshot data collection*. The *snapshot data collection capacity* is defined as $\Upsilon = \frac{nb}{\tau}$, where τ is the time used to collect all the data of a snapshot to the sink, i.e. snapshot data collection capacity reflects the average data receiving rate at the sink during snapshot data collection. Similarly, the process to collect all the data of N continuous snapshots is called *continuous data collection*. The *continuous data collection capacity* is defined as $\Upsilon = \frac{Nnb}{\tau}$, where τ now is the time consumption to collection all the data of these N snapshots to the sink, i.e. continuous data collection capacity reflects the average data receiving rate at the sink during continuous data collection. In this part, we study the snapshot data collection and continuous data collection problems for WSNs, as well as their achievable network capacities[1].

2.2.2 Routing Tree

Let $G(V, E)$ be a unit-disk graph representing a WSN. We define the sink s_0 as the *center* of G. The *radius* of G with respect to s_0 is the maximum depth of the *Breadth-First-Search* (BFS) tree rooted at s_0. For a subset U of V, U is a *Dominating Set* (DS)

[1] In the following of this part, we use snapshot/continuous data collection capacity and network capacity interchangeably without confusion.

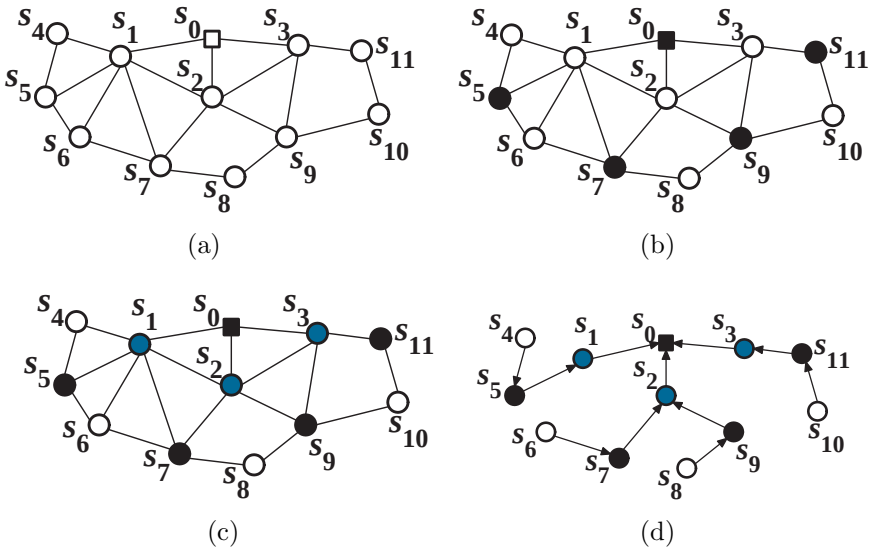


Figure 2.1 The construction of a CDS based routing tree. s_0 is the sink. The black nodes in (b) are *dominators*, and the blue nodes in (c) are *connectors*.

of G if every node in V is either an element of U or adjacent[2] to at least one node in U. If the subgraph of G induced by U is connected, then U is called a *Connected Dominating Set* (CDS) of G. Since CDS can serve as a virtual backbone of a WSN, it receives a lot of attention [7], [69]-[70], [71]-[72], recently.

Taking the WSN shown in Figure 2.1(a) as an example, we build a CDS based routing tree T (shown in Figure 2.1(d)) using the method proposed in [7]. Let G represent the network in Figure 2.1(a). T is rooted at sink s_0 and can be built according to the following steps. First, construct a Breadth-First-Searching (BFS) tree on G beginning at the sink and obtain a *Maximal Independent Set* (MIS) D according to the search sequence. As shown in Figure 2.1(b), the set of all the black nodes $\{s_0, s_5, s_7, s_9, s_{11}\}$ is a MIS of the network shown in Figure 2.1(a). Note that D is also a DS of G and an element in D is called a *dominator*. Clearly, every dominator is out of the communication range of any other dominators. Let G' be a graph on D in which two nodes in D linked by an edge if and only if these two nodes have a common neighbor in G, e.g. s_0 and s_7. Obviously, sink s_0 is in G' and we also denote

[2]In this part, if we say two nodes u and v are adjacent/connected, we mean u and v are within the communication range of each other, i.e. $\|u - v\| \leq 1$.

s_0 as the center of G'. Suppose that the radius of G' with respect to s_0 is L' and we denote the union of dominators at level l ($0 \leq l \leq L'$) as set D_l. Note that, $D_0 = \{s_0\}$. Second, we choose nodes, also called *connectors*, to connect all the nodes in D to form a CDS. Let S_l ($0 \leq l \leq L'$) be the set of the nodes adjacent to at least one node in D_l and at least one node in D_{l+1} and compute a minimal cover $C_l \subseteq S_l$ for D_{l+1}. Let $C = \cup_0^{L'-1} C_l$ and therefore $D \cup C$ is a CDS of G. As shown in Figure 2.1(c), the blue nodes $\{s_1, s_2, s_3\}$ are connectors chosen to connect the dominators in $D_0 = \{s_0\}$ and $D_1 = \{s_5, s_7, s_9, s_{11}\}$. Meanwhile, the union of the dominators and connectors in Figure 2.1(c) forms a CDS of the network shown in Figure 2.1(a). Finally, for any other node u, also called a *dominatee*, not belonging to $D \cup C$, choose the nearest dominator as u's parent node. In this way, the routing tree T of G is obtained as shown in Figure 2.1(d).

For each link in T, we assign it a direction from the child node to the parent node along the data transmission flow to the sink as shown in Figure 2.1(d). Furthermore, the receiving (respectively, transmitting) node, i.e. parent (respectively, child) node, of a link is called a *head* (respectively, *tail*). Suppose that A is a set of links of T. The corresponding *conflicting graph* of A is denoted by $\Re(A) = (V_A, E_A)$, where each link in A is abstracted to a node in V_A and two nodes in V_A form an edge in E_A if the corresponding two links of these two nodes are interfering links.

Lemma 2.2.1 in [7] can be used to derive some useful results of the routing tree T.

Lemma 2.2.1 [7] *Suppose that O (respectively, O') is a disk (respectively, half-disk) with radius r, and U is a set of points with mutual distances of at least one. Then the number of the points α_r in a disk and the number of the points β_r in a half-disk are*

$$\alpha_r = |U \cap O| \leq \frac{2\pi}{\sqrt{3}} r^2 + \pi r + 1 \tag{2.1}$$

$$\beta_r = |U \cap O'| \leq \frac{\pi}{\sqrt{3}} r^2 + (\frac{\pi}{2} + 1) r + 1. \tag{2.2}$$

From Lemma 2.2.1, the authors in [7] derived the following properties of the routing tree T. First, for each $0 \leq l \leq L' - 1$, each connector in C_l is adjacent to at most 4 dominators

in D_{l+1}. Second, for each $1 \leq l \leq L' - 1$, each dominator in D_l is adjacent to at most 11 connectors in C_l. Third, $|C_0| \leq 12$.

2.2.3 Vertex Coloring Problem

For a graph $G = (V, E)$, the *maximum degree* (respectively, *minimum degree*) of G is denoted by $\Delta(G)$ (respectively, $\delta(G)$). A subgraph of G on $U \subseteq V$ is denoted by $G(U)$. The *inductivity* of G is defined as $\delta^*(G) = \max_{U \subseteq V} \delta(G(U))$. A *vertex coloring* of G is a scheme of coloring all the vertices in G such that no two adjacent vertices share the same color. The *chromatic number* $\chi(G)$ of G is the least number of colors used to color G. Deciding the lower bound of $\chi(G)$ is a well-known NPC problem. However, the upper bound of $\chi(G)$ has been derived in *graph theory* [7][73]. The following lemma was proven in [7] and [73].

Lemma 2.2.2 *$\chi(G) \leq 1 + \delta^*(G)$ and a vertex coloring scheme, called* first-fit coloring*, for G using at most $1 + \delta^*(G)$ colors can be found in polynomial time.*

Given a link set A of T, the channel assignment problem for A can be abstracted to the vertex coloring problem for its corresponding conflicting graph $\Re(A)$. If the tail (respectively, head) of every link in A is a dominator, then Lemma 2.2.3 in [7] gives the upper bound of $\delta^*(A)$.

Lemma 2.2.3 [7] *$\delta^*(A) \leq \beta_{\rho+1} - 1$.*

Lemma 2.2.3 implies that in the worst case, at most $\beta_{\rho+1}$ channels may be assigned to all the links in A without channel interference by a first-fit coloring method.

2.3 Capacity of Snapshot Data Collection

In this section, we investigate the traditional snapshot data collection problem, propose a scheduling algorithm for this problem in single-radio multi-channel WSNs and analyze the achievable capacity of the proposed algorithm. Subsequently, we point out that the proposed

algorithm and most existing works cannot improve the capacity of a network by the pipeline technology.

Since at any time slot, the sink can receive data from at most one neighboring sensor, therefore, the upper bound of the snapshot data collection capacity is W [38][4][5]. Aiming at this upper bound, we design a scheduling algorithm for snapshot data collection which is order-optimal and has a tighter lower bound than the previously best result [4].

2.3.1 Scheduling Algorithm for Snapshot Data Collection

The idea of *single-path scheduling* has been employed in [35] and [4] to collect data for a WSN. However, their methods have a looser bound of the snapshot data collection capacity. In this subsection, we design a new *multi-path scheduling* algorithm based on the routing tree T built in Section 2.2, which is proven to have a better performance. We first study how to schedule a single path and then extend it to the scheduling of multi-path in the routing tree T.

For simplicity, we introduce the concept of *round*. A *round* is a period of time which consists of multiple continuous time slots. We take the path shown in Figure 2.2(a) as an example to explain the idea of the single path scheduling scheme. In Figure 2.2(a), the path, denoted by P, consists of one sink s_0 and three sensors s_1, s_2, and s_3, where s_0 and s_2 are dominators, s_1 is a connector, and s_3 is a dominatee. The value marked in each node is the number of the packets at this node to be transmitted during a time slot. Initially, every sensor on P has one packet and there is no packet at s_0. P_o (respectively, P_e) denotes the set of links on P whose heads (respectively, tails) are dominators and whose tails have at least one packet to be transmitted. For the path shown in Figure 2.2(a), $P_o = \{(s_3, s_2), (s_1, s_0)\}$ and $P_e = \{(s_2, s_1)\}$. We schedule P according to the following two steps and repeat them until all the packets have been collected by s_0.

Step 1: In an odd round, schedule every link in P_o once, i.e. assign a dedicated channel and one dedicated time slot to each link in P_o.

Step 2: In an even round, schedule every link in P_e once, i.e. assign a dedicated channel

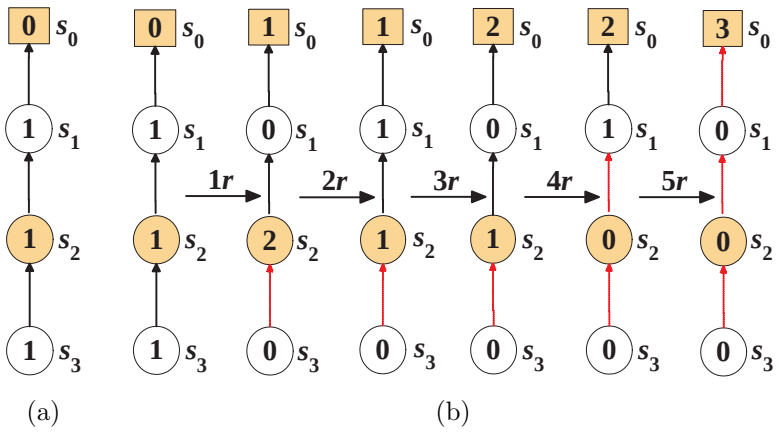


Figure 2.2 (a) A single path and (b) its scheduling (*r=round*).

and one dedicated time slot to each link in P_e.

The detailed scheduling in Step 1 can be conducted in the following way: first, for any link $\iota_i \in P_o$, let $\mathbb{I}_{P_o}(\iota_i) = \{\iota_j | \iota_j \in P_o,\ \iota_i \text{ and } \iota_j \text{ are interfering links}\}$; second, sort the links in P_o according to $|\mathbb{I}_{P_o}(\iota_i)|$ $(1 \leq i \leq |P_o|)$ in a non-decreasing order, where $|\cdot|$ denotes the cardinality of a set, and denote the resulting link sequence as $\{\iota'_1, \iota'_2, \ldots, \iota'_{|P_o|}\}$; finally, during the i-th $(1 \leq i \leq \left\lceil \frac{|P_o|}{H} \right\rceil)$ time slot of a round, let the j-th $((i-1)H < j \leq iH)$ link in P_o work on channel $\lambda_{j\%H+1}$. Here, we sort the links in P_o first and subsequently assign channels is based on the *first-fit coloring* scheme in Lemma 2.2.2. Furthermore, according to Lemma 2.2.2 and Lemma 2.2.3, the channel assignment plan for the links in P_o is interference/collision-free. The detailed scheduling in Step 2 is similar to that of Step 1.

The scheduling process of P in Figure 2.2(a) is shown in Figure 2.2(b). During the first (odd) round, links (s_3, s_2) and (s_1, s_0) are scheduled and the packets at s_3 and s_1 are transmitted to their parent nodes. After the first round, s_3 has no packet to transmit. During the second (even) schedule, link (s_2, s_1) is scheduled and s_2 transmits one packet to its parent node. This process continues until all the packets on path P has been transmitted to s_0.

We now consider the scheduling of the routing tree T built in Section 2.2. Suppose that there are m leaf nodes in T denoted by $s_1^l, s_2^l, \cdots, s_m^l$ respectively. The path from leaf node

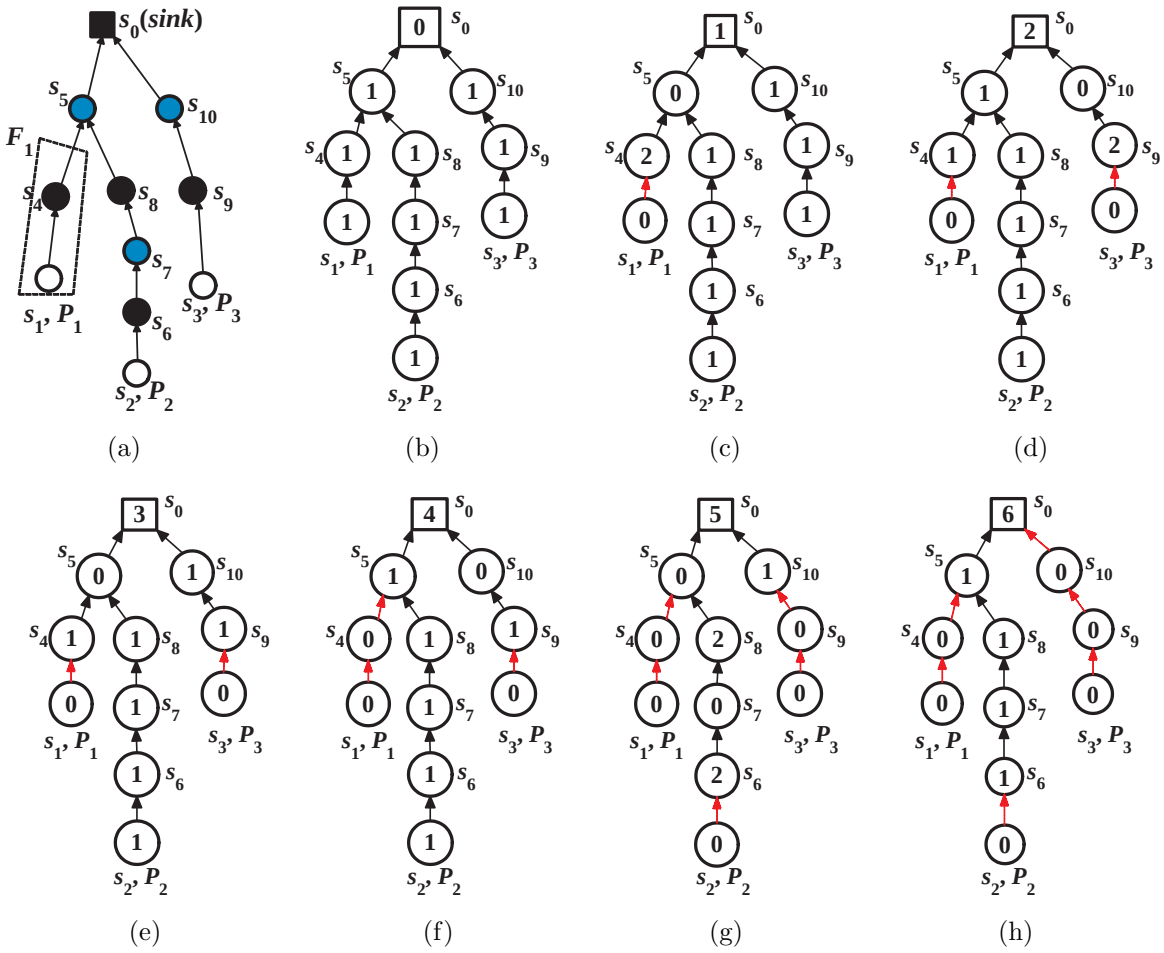


Figure 2.3 A routing tree and its scheduling. In (a), black nodes are dominators, blue nodes are connectors, and the other nodes are dominatees.

s_i^l $(1 \leq i \leq m)$ to the sink s_0 is denoted by P_i. Two paths P_i and P_j are said *intersecting* if they have at least one common node besides the sink node. Assume path P_i and P_j are intersecting, the lowest common ancestor of s_i^l (P_i) and s_j^l (P_j), i.e. the common node of P_i and P_j having the largest number of hops from the sink, is called an *intersecting point* of P_i and P_j. If path P_i intersects with other paths, the route from s_i^l to the nearest intersecting point of P_i is called a *sub-path*, denoted by F_i. Otherwise, F_i is actually P_i.

Taking the routing tree $\widehat{T}$ shown in Figure 2.3(a) as an example, $\widehat{T}$ consists of one sink s_0 and 10 sensor nodes denoted by s_i $(1 \leq i \leq 10)$. $\widehat{T}$ has three leaf nodes s_1, s_2, and s_3, which correspond to paths P_1, P_2, and P_3, respectively. In $\widehat{T}$, P_1 and P_2 are intersecting

and their intersecting point is s_5. Nevertheless, P_1 and P_3, as well as P_2 and P_3, are not intersecting since they have no common node beside s_0. For P_1, the route from s_1 to s_5 is the sub-path of P_1, denoted by F_1. For P_3, since it is not intersecting with any path, F_3 is P_3 itself.

Algorithm 1: Multi-path Scheduling Algorithm

input : a routing tree T with m leaf nodes
output: a schedule plan for the routing tree T

```
1  for i = 1; i ≤ m; i++ do
2      while there is some data for transmission on F_i do
3          P ← {P_i};
4          S ← ∅;
5          if rd_i%2 == 1 then
6              S ← P_o^i;
7              rd_i++;
8          else if rd_i%2 == 0 then
9              S ← P_e^i;
10             rd_i++;
11         for j = i + 1; j ≤ m; j++ do
12             if P_j is not intersecting with any path in P && there is some data for
               transmission on F_j then
13                 if rd_j%2 == 1 && all the transmissions in P_o^j and all the
                   transmissions in S are interference/collision-free then
14                     P ← P ∪ {P_j};
15                     S ← S ∪ P_o^j;
16                     rd_j++;
17                 if rd_j%2 == 0 && all the transmissions in P_e^j and all the
                   transmissions in S are interference/collision-free then
18                     P ← P ∪ {P_j};
19                     S ← S ∪ P_e^j;
20                     rd_j++;
21         schedule the links in S in a round as in the single-path scheduling algorithm;
22     if there is no data for transmission on F_i then
23         remove all the links on F_i from T;
```

To schedule multiple paths on the routing tree T, we propose a *multi-path scheduling* algorithm as shown in Algorithm 1. In Algorithm 1, $\mathcal{P}$ is the set of paths been scheduled

simultaneously in a round, $\mathcal{S}$ is the set of links from multiple paths that can be scheduled in a round, rd_i/rd_j indicates the number of available rounds that has been assigned to path P_i/P_j, and P_o^i/P_o^j (respectively, P_e^i/P_e^j) is the set of links on P_i/P_j whose heads (respectively, tails) are dominators and whose tails have at least one packet to be transmitted. From Algorithm 1, we can see that lines 2-10 are used to schedule path P_i according to the single-path scheduling algorithm. Lines 11-20 are used to find other paths that can be scheduled simultaneously with P_i according to the single-path scheduling algorithm at the same round.

We further explain the multi-path scheduling algorithm through the routing tree $\widehat{T}$ shown in Figure 2.3(a). Assume the interference radius $\rho = 1$, i.e. the interference radius is equal to the transmission radius, which implies each *round* consists of two time slots. Furthermore, we use $\mathcal{I}(s_i)$ $(1 \leq i \leq n)$ to denote the set of sensor nodes that cannot be transmitted data simultaneously with s_i. For the nodes in $\widehat{T}$, we assume $\mathcal{I}(s_1) = \{s_4, s_5, s_6, s_7, s_8\}$, $\mathcal{I}(s_2) = \{s_6, s_7\}$, $\mathcal{I}(s_3) = \{s_9, s_{10}\}$, $\mathcal{I}(s_4) = \{s_1, s_5, s_7, s_8\}$, $\mathcal{I}(s_5) = \{s_1, s_4, s_7, s_8, s_{10}\}$, $\mathcal{I}(s_6) = \{s_1, s_2, s_7, s_8\}$, $\mathcal{I}(s_7) = \{s_1, s_2, s_4, s_5, s_6, s_8\}$, $\mathcal{I}(s_8) = \{s_1, s_4, s_5, s_6, s_7\}$, $\mathcal{I}(s_9) = \{s_3, s_{10}\}$, and $\mathcal{I}(s_{10}) = \{s_3, s_5, s_9\}$. Additionally, for path P_1, $P_o^1 = \{(s_1, s_4), (s_5, s_0)\}$ and $P_e^1 = \{(s_4, s_5)\}$, for path P_2, $P_o^2 = \{(s_2, s_6), (s_7, s_8), (s_5, s_0)\}$ and $P_e^2 = \{(s_6, s_7), (s_8, s_5)\}$, and for path P_3, $P_o^3 = \{(s_3, s_9), (s_{10}, s_0)\}$ and $P_e^3 = \{(s_9, s_{10})\}$. At the beginning of Algorithm 1, the network is shown in Figure 2.3(b) with the number inside each node denoting the number of the data packets at this node. According to the algorithm, during the first round, P_o^1 is scheduled, and path P_2 will not be scheduled since it is intersecting with P_1. P_3 also will not be scheduled since the link (s_{10}, s_0) in P_o^3 and the link (s_5, s_0) in P_o^1 are not interference/confliction-free. Thus, after the first round, the network situation is shown in Figure 2.3(c). During the second round, P_e^1 will be scheduled. Now, all the links in P_e^1 and all the links in P_o^3 are conflict/interference-free (Here, we consider P_o^3 instead of P_e^3 is because for path P_3, the current round is the first available round.). Hence, P_1 and P_3 can be scheduled simultaneously at the second round. After the second round, the network situation is shown in Figure 2.3(d). Similarly, according to Algorithm 1, the network after the third, the fourth, the fifth, and the sixth round is shown in Figure 2.3(e), (f), (g), and

(h), respectively. Finally, for $\widehat{T}$ shown in Figure 2.3(a), it will take 13 rounds to collect all the data packets to the sink by the multi-path scheduling algorithm. By contrast, it will take 18 rounds to collect all the data packets to the sink by the single-path scheduling algorithm.

2.3.2 Capacity Analysis

In this subsection, we analyze the achievable network capacity of the proposed multi-path scheduling algorithm. The upper bound of the snapshot data collection capacity is W which has been explained. Consequently, we focus on the lower bound of the snapshot data collection capacity. In the worst case, all the paths in the routing tree T are intersecting, i.e. they have a common intersecting point, which means only one path can be scheduled at any time. In order to derive the lower bound of the multi-path scheduling algorithm, we first investigate the number of the rounds needed to finish the scheduling of one single path and then study the number of the time slots in each round. Lemma 2.3.1 gives the maximum number of the rounds used for the scheduling of one single path.

Lemma 2.3.1 *For a single path P of length L in T, it takes at most $2L-1$ rounds to collect all the packets on P at the sink node.*

Proof: Suppose that the node sequence on P is $s_1, s_2, \ldots, s_L, s_0$, where s_1 is the leaf node (dominatee), and s_0 is the sink node. Considering the building process of T, each link in P has either a dominator head or a dominator tail. According to the scheduling scheme of a single path, during the first (odd) round, the links in P_o are scheduled, which implies each non-dominator with at least one packet transmits this packet to its parent node. After the first round, the sink, receives one packet and all the other dominators of the links in P_o have two packets to be transmitted. During the second (even) round, the links in P_e are scheduled, which implies that every dominator in P_e transmits one packet to its parent node. As a result, the sensor s_i $(2 \leq i \leq L)$ has exactly one packet to be transmitted and a new odd-even scheduling round begins. In summary, after every two rounds, the sink receives one packet and the length of the data collection path decreases by 1. Since the length of P

is L and s_0 is the destination of all the packets which does not have to transmit any data, it takes at most $2L-1$ rounds to collect all the packets on P. □

From Lemma 2.3.1, it is straightforward to obtain the number of the rounds used to collect all the data on the sub-path F of P as shown in Corollary 2.3.1.

Corollary 2.3.1 *For the sub-path F of length L_s in P , it takes at most $2L_s$ rounds to collect all the packets on F.*

Proof: The proof of Corollary 2.3.1 is similar to that of Lemma 2.3.1. Note that the intersecting point is not a sink node in this case and thus it needs one round to transmit its packet. □

By Lemma 2.3.1 and Corollary 2.3.1, we can obtain the number of the rounds used to collect the packets on a path. The maximum number of the time slots in a round is as follows.

Lemma 2.3.2 *In the single-path scheduling algorithm, a round has at most $\left\lceil \frac{\beta_{\rho+1}}{H} \right\rceil$ time slots, where $\beta_{\rho+1}$ is the number of the dominators in a half-disk with radius $\rho+1$ and H is the number of available orthogonal channels.*

Proof: During every odd (respectively, even) round, the scheduled links are links in P_o (respectively, P_e). Since the heads (respectively, tails) of links in P_o (respectively, P_e) are dominators, we can schedule all the links in P_o (respectively, P_e) in one time slot with at most $\beta_{\rho+1}$ channels in polynomial time by Lemma 2.2.3 and Lemma 2.2.2. Now, we have H available channels, which means we can finish the scheduling within $\left\lceil \frac{\beta_{\rho+1}}{H} \right\rceil$ time slots. Therefore, the lemma holds. □

Now we can obtain the lower bound of the achievable capacity of the multi-path scheduling algorithm as shown in Theorem 2.3.1.

Theorem 2.3.1 *The capacity Υ at the sink of T of the multi-path scheduling algorithm is at least $\frac{W}{2\lceil (1.81\rho^2+c_1\rho+c_2)/H \rceil}$, where $c_1 = \frac{2\pi}{\sqrt{3}} + \frac{\pi}{2} + 1$ and $c_2 = \frac{\pi}{\sqrt{3}} + \frac{\pi}{2} + 2$, which is order-optimal.*

Proof: Suppose that T has m paths and the length of each path is L_i $(1 \leq i \leq m)$. In the worst case, all the m paths cannot be scheduled concurrently. Then by Lemma 2.3.1, Corollary 2.3.1 and Lemma 2.3.2, the total time τ used to collect all the packets of T at the sink is at most $t \cdot \sum_{i=1}^{m} 2L_i \left\lceil \frac{\beta_{\rho+1}}{H} \right\rceil$. According to the multi-path scheduling algorithm, for any path P_i, the time used to collect packets on P_i is equal to the time used to collect packets on the corresponding sub-path F_i of P_i[3]. Therefore, $\tau \leq t \cdot \sum_{i=1}^{m} 2L_i \left\lceil \frac{\beta_{\rho+1}}{H} \right\rceil = t \cdot \sum_{i=1}^{m} 2\left|F_i\right| \left\lceil \frac{\beta_{\rho+1}}{H} \right\rceil = 2t \left\lceil \frac{\beta_{\rho+1}}{H} \right\rceil \sum_{i=1}^{m} |F_i|$.

Since the number of the links in T is equal to the number of the sensors in T, $\sum_{i=1}^{m} |F_i| = n$. Then, $\tau \leq 2nt \left\lceil \frac{\beta_{\rho+1}}{H} \right\rceil$. Therefore, the capacity

$$\Upsilon = \frac{nb}{\tau} \geq \frac{nb}{2nt \left\lceil \frac{\beta_{\rho+1}}{H} \right\rceil} = \frac{b}{2t \left\lceil \frac{\beta_{\rho+1}}{H} \right\rceil} = \frac{W}{2 \left\lceil \frac{\beta_{\rho+1}}{H} \right\rceil}. \tag{2.3}$$

From, Lemma 2.2.1, we have

$$\beta_{\rho+1} \leq \frac{\pi}{\sqrt{3}}(\rho+1)^2 + (\frac{\pi}{2}+1)(\rho+1) + 1 \tag{2.4}$$

$$= \frac{\pi}{\sqrt{3}}\rho^2 + (\frac{2\pi}{\sqrt{3}} + \frac{\pi}{2} + 1)\rho + \frac{\pi}{\sqrt{3}} + \frac{\pi}{2} + 2 \tag{2.5}$$

$$\approx 1.81\rho^2 + c_1\rho + c_2, \tag{2.6}$$

where $c_1 = \frac{2\pi}{\sqrt{3}} + \frac{\pi}{2} + 1$ and $c_2 = \frac{\pi}{\sqrt{3}} + \frac{\pi}{2} + 2$. This implies $\Upsilon \geq \frac{W}{2\left\lceil \frac{\beta_{\rho+1}}{H} \right\rceil} \geq \frac{W}{2\left\lceil \frac{1.81\rho^2 + c_1\rho + c_2}{H} \right\rceil}$. Since H is a constant and the upper bound of Υ is W, Υ is order-optimal. □

From Theorem 2.3.1, we know that the achievable capacity of the multi-path scheduling algorithm is order-optimal, and it also has a tighter lower bond compared with the previously best result in [4], which has a lower bound of $\frac{W}{8\rho^2}$.

[3]From lines 2-10 in Algorithm 1, the scheduling of path P_i is stopped when all the data packets on the sub-path F_i have been collected by the sink. As shown in Figure 2.3(f) and (g), after all the data packets on F_1 (the sub-path of P_1) have been collected by the sink, we begin to schedule path P_2. Additionally, based on the definition of a sub-path, F_3 in Figure 2.3 is P_3 itself since P_3 does not intersect with any path. Therefore, the time used to collect packets on P_i is equal to the time used to collect packets on the corresponding sub-path F_i of P_i.

2.3.3 Discussion

When we address the continuous data collection problem, an intuitive idea is to pipeline the existing snapshot data collection operations [4]. Nevertheless, such an idea cannot achieve a better performance. This is because the sink can receive at most one data packet at a time slot. By pipeline, data transmissions at the nodes far from the sink are really accelerated. However, the fact that a sink can receive at most one packet at each time slot makes the data accumulated at the nodes near the sink. Finally, the network capacity cannot be improved even with pipeline. This motivates us to investigate new methods for continuous data collection.

2.4 Capacity of Continuous Data Collection

Since multi-path scheduling algorithm and existing works with pipeline cannot improve the capacity of continuous data collection, we propose a novel *pipeline scheduling* algorithm based on *compressive data gathering* (CDG) [1] in dual-radio multi-channel WSNs, which augments the continuous data collection capacity significantly. Here we consider dual-radio multi-channel WSNs because dual radios can make a *half-duplex* single-radio node work in a *full-duplex* mode, i.e. a dual-radio node can receive and transmit data simultaneously with the two radios over different channels. Furthermore, the full-duplex working mode is in favor of pipeline. For completeness, we also analyze the achievable network capacity of the pipeline scheduling algorithm (a little modification is needed) in single-radio multi-channel WSNs.

2.4.1 Compressive Data Gathering (CDG)

CDG is first proposed in [1] for snapshot data collection in single-radio single-channel WSNs. The basic idea of CDG is to distribute the data collection load uniformly to all the nodes in the entire network. We take the data collection on a path consisting of L sensors $s_1, s_2, \ldots, s_L$ and one sink s_0 as shown in Figure 2.4 [1] as an example to explain CDG. In

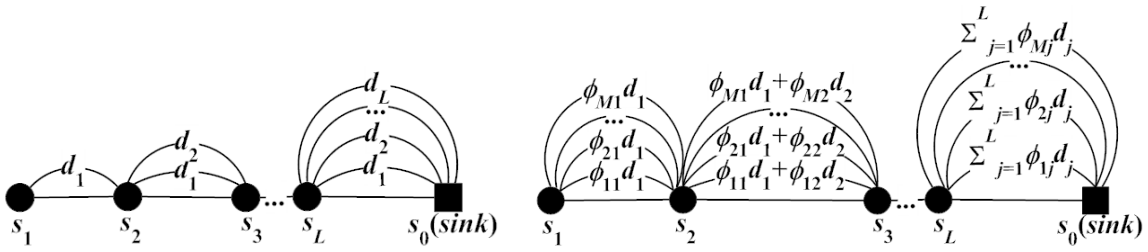


Figure 2.4 Comparing of (a) basic data collection and (b) CDG [1].

Figure 2.4, the packet produced at sensor s_j $(1 \leq j \leq L)$ is d_j. In the basic data collection shown in Figure 2.4(a), s_1 transmits one packet d_1 to s_2, s_2 transmits two packets d_1 and d_2 to s_3, and finally all the packets on the path are transmitted to s_0 by s_L. Obviously, nodes near the sink has more transmission load compared with nodes far from the sink in the basic data collection. To balance the transmission load, the authors in [1] proposed the CDG method as shown in Figure 2.4(b). Instead of transmitting the original data directly, s_1 multiplies its data with a random coefficient ϕ_{i1} $(1 \leq i \leq M)$, and sends the M results $\phi_{i1}d_1$ to s_2. Upon receiving $\phi_{i1}d_1$ $(1 \leq i \leq M)$ from s_1, s_2 multiplies its data d_2 with a random coefficient ϕ_{i2} $(1 \leq i \leq M)$, adds it to $\phi_{i1}d_1$, and then sends $\phi_{i1}d_1 + \phi_{i2}d_2$ as one data packet to s_3. Finally, s_L does the similar multiplication and addition and sends the result $\sum_{j=1}^{L} \phi_{ij}d_j$ $(1 \leq i \leq M)$ to s_0. After s_0 receives all the M packets, s_0 can restore the original packets based on the compressive sampling theory [1]. By CDG, all the sensors send M packets to their parent nodes, which achieves the goal to uniformly distribute the data collection task to the entire network. The number of the transmitted packets is $O(n^2)$ in Figure 2.4(a) and is $O(NM)$ in Figure 2.4(b), and usually $M \ll n$ for large scale WSNs. Therefore, CDG reduces the number of the transmitted packets.

2.4.2 Pipelining

In computing, a *pipeline* is a set of data processing elements connected in series, so that the output of one element is the input of the next one and the elements of a pipeline are often executed in parallel. For instance, Figure 2.5 shows a pipeline system consisting of

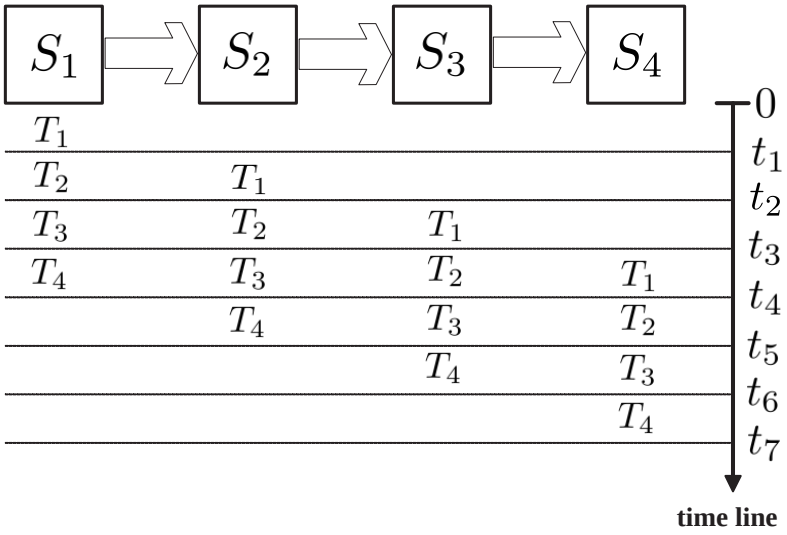


Figure 2.5 A pipeline system.

four functional element S_1, S_2, S_3, and S_4 to address four tasks T_1, T_2, T_3, and T_4. To finish four tasks by this pipeline, we can input these tasks sequentially for processing. As shown in Figure 2.5, we first input task T_1 (at time 0) to the functional element S_1 for processing. After T_1 is processed by S_1 (at time t_1), S_1 outputs the result to S_2 for processing, and meanwhile, T_2 will be input to S_1 (also at time t_1) for processing. Then, at some time slot, it can be achieved that multiple tasks are processed simultaneously at different elements of the pipeline system. For instance, all the four tasks are processed by the pipeline system during time slot (t_3, t_4) in Figure 2.5. Evidently, by exploiting the pipeline technique, the efficiency of the entire functional system can be improved and thus the time consumption to process multiple tasks can be decreased. Consequently, to improve the efficiency and reduce the induced delay of the data collection process of continuous data collection, we will partition the network into different functional elements to form an efficient data collection pipeline.

2.4.3 Pipeline Scheduling

Thanks to the benefit brought by CDG, we can address the continuous data collection problem with the pipeline technique. From the building process of the routing tree T, we know that the nodes in T can be divided into sets by levels $D_e, D_{L'}, C_{L'-1}, D_{L'-1}, C_{L'-2}, \ldots, D_1, C_0, D_0 =$

$\{s_0|s_0$ is the sink$\}$ in a bottom-up way, where D_e is the set of all the dominatees, i.e. leaf nodes, D_i $(0 \leq i \leq L')$ is the set of the dominators at the i-th level, and C_i $(0 \leq i \leq L'-1)$ is the set of the connectors at the i-th level. Since every node has two radios, one radio can be dedicated to receive data and the other dedicated to transmit data. Therefore, the nodes at every level can receive and transmit data simultaneously over different channels. Consequently, for a continuous data collection task consisting of N snapshots, we propose a *pipeline scheduling* algorithm as follows.

Step 1: The nodes at the dominatee level transit data packets to their parent nodes snapshot by snapshot in the CDG way. All the nodes in D_e transmit the packets of the j-th $(1 \leq j \leq N-1)$ snapshot to their parent nodes in the CDG way, i.e. for every node $s \in D_e$, s multiplies its data with M random coefficients respectively, and sends the M products to its parent node. After all the packets of the j-th snapshot have been transmitted successfully, the nodes in D_e immediately transmit the packets of the $(j+1)$-th snapshot in the CDG way.

Step 2: After the nodes at each dominator level receive all the data packets of the j-th snapshot, they transmit the data of the j-th snapshot to their parent nodes in the CDG way. After all the nodes in D_l $(1 \leq l \leq L')$ receive all the packets of the j-th snapshot from their child-level, they send the packets of the j-th snapshot to their parent nodes in the CDG way, i.e. every node $s \in D_l$ combines its packet of the j-th snapshot with the received packets of the j-th snapshot, and sends the M new packets to its parent node. After all the packets of the j-th snapshot have been transmitted successfully, the nodes in D_l immediately transmit the packets of the $(j+1)$-th snapshot to their parent nodes in the CDG way, if they have received all the packets of the $(j+1)$-th snapshot from their child-level.

Step 3: After the nodes at each connector level receive all the data packets of the j-th snapshot, they transmit the data of the j-th snapshot to their parent nodes in the CDG way. After all the nodes in C_l $(0 \leq l \leq L'-1)$ receive all the packets of the j-th snapshot from their child-level, they send the packets of the j-th snapshot to their parent nodes in the CDG way, i.e. every node $s \in C_l$ combines its packet of the j-th snapshot with the received

packets of the j-th snapshot, and sends the M new packets to its parent node. After all the packets of the j-th snapshot have been transmitted successfully, the nodes in C_l immediately transmit the packets of the $(j+1)$-th snapshot in the CDG way if they have received all the packets of the $(j+1)$-th snapshot from their child-level.

Step 4: The sink restores the data of a snapshot in the CDG way after it receives all the packets of this snapshot.

Steps 1-4 provide the general frame of our pipeline scheduling scheme. Now, we discuss how to prevent radio confliction and channel interference in Steps 1-3. If two or more nodes have the same parent node, we call them *sibling* nodes. In Steps 1-3, radio confliction may arise if two or more sibling nodes send data to their parent node simultaneously even over different orthogonal channels. This is because every sensor only has one radio dedicated to receiving data. Suppose that there are at most Δ_e (respectively, Δ_d and Δ_c) nodes in D_e (respectively, D_l $(1 \leq l \leq L')$ and C_l $(1 \leq l \leq L'-1)$) which have the same parent node. Usually, $\Delta_e < \Delta(T)$ except in one-hop WSNs, where any sensor is just one hop away from the sink, $\Delta_e = \Delta(T)$. Then, $\Delta_d \leq 4$ and $\Delta_c \leq 11$ (Note that $|C_0| \leq 12$.) (see Section 2.2.2). To avoid confliction, we divide the nodes in D_e (respectively, D_l $(1 \leq l \leq L')$ and C_l $(1 \leq l \leq L'-1)$) into Δ_e (respectively, Δ_d and Δ_c) subsets to guarantee that each node belongs to one subset and no sibling nodes belong to the same subset. Then, when we schedule the nodes of each level, we schedule these subsets in a certain order. For the nodes in C_0, we schedule them in a certain order, e.g. the nodes with small IDs are scheduled with high priority.

Different from the multi-path scheduling algorithm, in which a sensor sends one packet over a link in one time slot, we employ the CDG way, where a sensor sends M packets for a snapshot. We now introduce the concept of a *Super Time Slot* (STS) which consists of M time slots. In a STS, a sensor can send M packets over a channel for a snapshot. For the links working simultaneously, we assign channels and STSs in the similar way of the multi-path scheduling algorithm.

2.4.4 Capacity Analysis

In this subsection, we analyze the achievable network capacity of the proposed *pipeline scheduling* algorithm. For completeness, we also analyze the achievable network capacity of the pipeline scheduling algorithm in single-radio multi-channel WSNs (a little modification is needed since each sensor has one radio now) at the end of this subsection.

Lemma 2.4.1 indicates the inductivity (defined in Section 2.2.3) of the corresponding conflicting graph of the links scheduled simultaneously in the pipeline scheduling algorithm, which is used to obtain the upper bound of the number of the necessary channels to schedule these links.

Lemma 2.4.1 *Suppose that A is the set of the links in T scheduled simultaneously in the pipeline scheduling algorithm, and $\Re(A)$ is the corresponding conflicting graph of A, then, $\delta^*(\Re(A)) \leq 2\beta_{\rho+2} - 1$, where $\delta^*(\Re(A))$ is the inductivity of $\Re(A)$ and $\beta_{\rho+2}$ is the number of the dominators within a half-disk of radius $\rho + 2$.*

Proof: Since the sibling nodes at every level have been divided into different subsets and different subsets are scheduled in a certain order, there is no radio confliction among the links in A. Furthermore, for any link in A, either the tail or the head of this link is a dominator according to the building process of the routing tree T. Suppose that A' is a subset of A and e is the link in A' whose tail, denoted by $t(e)$, or head, denoted by $h(e)$, is the bottommost dominator among all the dominators in A'. Then, we prove the number of the links interfered with e, i.e. $\delta(\Re(A'))$, is at most $2\beta_{\rho+2} - 1$ case by case as follows.

Case 1: $t(e)$ is a dominator. In this case, assume that e' is another link in A' interfered with e and $t(e')$ is a dominator. Since $t(e)$ is the bottommost dominator, the necessary condition for e and e' to be interfering links is that $t(e')$ locates at the upper half-disk centered at $t(e)$ with radius $\rho + 1$. On the other hand, if $h(e')$ is a dominator, then the necessary condition for e and e' to be interfering links is that $h(e')$ locates at the upper half-disk centered at $t(e)$ with radius $\rho + 2$. By Lemma 2.2.1, the number of the dominators within a half-disk of radius $\rho + 2$ is at most $\beta_{\rho+2}$. Since every dominator in A' is associated

with at most two links, there are at most $2\beta_{\rho+2}$ links within the half-disk of radius $\rho+2$ centered at $t(e)$. Therefore, $\delta(\Re(A')) \leq 2\beta_{\rho+2}-1$, where minus 1 means e is also in the half-disk. As a result, $\delta^*(\Re(A)) = \max_{A' \subseteq A} \delta(\Re(A')) \leq 2\beta_{\rho+2}-1$.

Case 2: $h(e)$ is a dominator. By the similar method as in Case 1, it can be proven that the conclusion also holds in this case. □

Based on the result of Lemma 2.4.1, we can determine the number of the STSs used to schedule all the links in A as follows.

Lemma 2.4.2 *For the links of set A in Lemma 2.4.1, we can use $\left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil$ STSs to schedule them without channel interference.*

Proof: By Lemma 2.4.1, $\delta^*(\Re(A)) \leq 2\beta_{\rho+2}-1$. By Lemma 2.2.2, we can use $1+\delta^*(\Re(A)) \leq 2\beta_{\rho+2}$ channels to schedule all the links in A in one STS simultaneously. Now, we have H channels, which implies we can schedule all the links in A in $\left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil$ STSs of H links in each STS. □

From the pipeline scheduling algorithm we know that the transport of subsequent snapshots has some time overlap with the transport of preceding snapshots. Therefore, we first analyze the time used to collect the packets of the first snapshot since it is the base of the pipeline, and then analyze the achievable capacity of the entire pipeline.

Theorem 2.4.1 *The number of the time slots used to collect the packets of the first snapshot by the pipeline scheduling algorithm is at most $M \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil (\Delta_e + 15L' + 1)$.*

Proof: In Step 1 of the pipeline scheduling algorithm, we divide the nodes in D_e into Δ_e subsets and schedule them in a certain order. By Lemma 2.4.2, each scheduling uses at most $\left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil$ STSs. Consequently, Step 1 needs at most $\Delta_e \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil$ STSs to finish the scheduling for the first snapshot. In Step 2 (respectively, Step 3), we divide the nodes in D_l $(1 \leq l \leq L')$ (respectively, C_l $(1 \leq l \leq L'-1)$) into Δ_d (respectively, Δ_c) subsets and schedule them in a certain order. Since, $\Delta_d \leq 4$ (respectively, $\Delta_c \leq 11$), Step 2 (respectively, Step 3) needs at most $4L' \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil$ (respectively, $11(L'-1) \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil$) STSs to finish the scheduling for the first

snapshot. Furthermore, it needs at most $12\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil$ STSs for C_0 to transmit the packets for the first snapshot to the sink. In summary, the total number of the STSs used for the first snapshot is at most

$$\Delta_e\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil + 4L'\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil + 11(L'-1)\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil + 12\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil \tag{2.7}$$

$$= \left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil(\Delta_e + 4L' + 11L' - 11 + 12) \tag{2.8}$$

$$= \left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil(\Delta_e + 15L' + 1). \tag{2.9}$$

Since every STS has M time slots, then the number of the time slots used for the first snapshot is at most $M\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil(\Delta_e + 15L' + 1)$. □

On the basis of the result in Theorem 2.4.1, we obtain the time slots used to collect all the packets of N continuous snapshots for the pipeline scheduling algorithm as shown in Theorem 2.4.2.

Theorem 2.4.2 *The time slots used for the pipeline scheduling algorithm to collect N continuous snapshots are at most $M\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil(\Delta_e + 15L' + 12N - 11)$ when $\Delta_e \leq 12$ or $M\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil(N\Delta_e + 15L' + 1)$ when $\Delta_e > 12$.*

Proof: From the proof of Theorem 2.4.1 we know, it takes the nodes in D_e (respectively, D_l $(1 \leq l \leq L')$, C_l $(1 \leq l \leq L'-1)$ and C_0) at most $\Delta_e\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil$ (respectively, $4\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil$, $11\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil$ and $12\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil$) STSs to transmit packets for a snapshot. In order to obtain the upper bound of the number of the time slots used, we assume the STSs used by nodes in D_e (respectively, D_l $(1 \leq l \leq L')$, C_l $(1 \leq l \leq L'-1)$ and C_0) are $\Delta_e\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil$ (respectively, $4\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil$, $11\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil$ and $12\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil$) in the following proof. Then, we prove Theorem 2.4.2 by cases.

Case 1: $\Delta_e \leq 4$. For clearness, we use the transmission of two snapshots S-1 and S-2 shown in Figure 2.6(a) as an example for explanation. In Figure 2.6(a), the vertical axis denotes the levels in the routing tree T and the horizontal axis denotes time slots. $t_e = \Delta_e\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil$, $t_d = 4\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil$, $t_c = 11\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil$, and $t_0 = 12\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil$, respectively. From

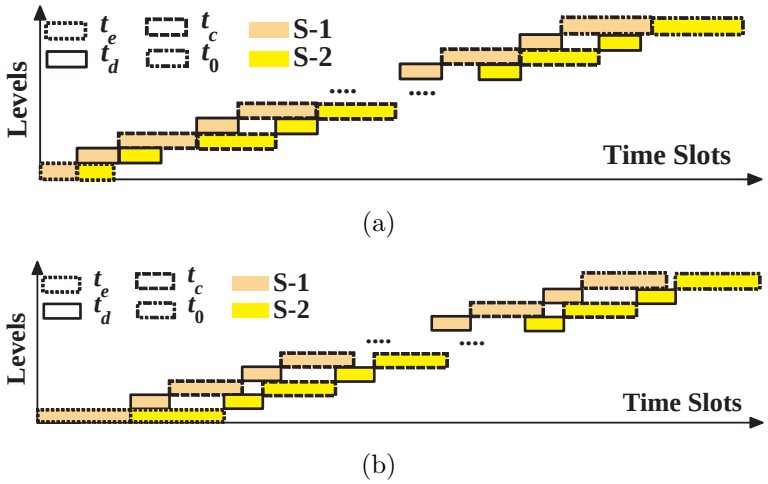


Figure 2.6 Data transport in (a) Case 1 and (b) Case 4.

Figure 2.6(a) we know, the nodes at the D_e-level begin to send packets of S-2 immediately after they send out the packets of S-1. Since $\Delta_e \leq 4$, after the nodes at the $D_{L'}$-level receive all the packets of S-2, they may still be busy with the transmission of the packets of S-1. Nevertheless, from the $C_{L'-1}$-level to the D_1-level, the pipeline can be utilized in a maximum degree, which implies whatever the packets of S-1 or the packets of S-2, they can be sent immediately. After the packets of S-2 are sent from the nodes at the D_0-level to the nodes in C_0, they may have to wait for a while at the nodes of the C_0-level, since the transmission for the packets of S-1 may last for as long as $12 \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil$ STSs. This implies the sink will receive all the packets of S-2 in $12 \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil$ STSs after it receives all the packets of S-1. According to the description of the pipeline scheduling algorithm, the subsequent snapshots will be transmitted in the same way, which implies the sink will receive all the packets of a snapshot within at most every $12 \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil$ STSs, after it receives the packets of the first snapshot which takes at most $M \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil (\Delta_e + 15L' + 1)$ time slots by Theorem 2.4.1. As a result, the number of time slots used to collect the packets of N continuous snapshots by the pipeline scheduling

algorithm is at most

$$M\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil(\Delta_e+15L'+1)+(N-1)\cdot 12M\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil \tag{2.10}$$

$$=M\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil(\Delta_e+15L'+1+12N-12) \tag{2.11}$$

$$=M\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil(\Delta_e+15L'+12N-11). \tag{2.12}$$

Case 2: $4<\Delta_e\leq 11$ *and Case 3:* $\Delta_e=12$. These two cases can be proven by the similar method used in Case 1. The number of the time slots used to collect the packets of N continuous snapshots is also at most $M\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil(\Delta_e+15L'+12N-11)$.

Case 4: $\Delta_e>12$. We use the data transmission of two snapshots shown in Figure 2.6(b) as an example to show the proof. The notations in Figure 2.6(b) are the same as those in Figure 2.6(a). Since $\Delta_e>12$, the pipeline can be utilized in a maximum degree at the $D_{L'}$-level and continue to the C_0-level. Then, the sink can receive all the packets of a subsequent snapshot every $\Delta_e\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil$ STSs after it receives the packets of the first snapshot. Therefore, the number of the time slots used to collect the packets of N continuous snapshots is at most

$$M\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil(\Delta_e+15L'+1)+(N-1)\cdot\Delta_e M\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil \tag{2.13}$$

$$=M\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil(\Delta_e+15L'+1+(N-1)\Delta_e) \tag{2.14}$$

$$=M\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil(N\Delta_e+15L'+1). \tag{2.15}$$

As a conclusion, Theorem 2.4.2 is true. □

Theorem 2.4.2 shows the number of the time slots used to collect N continuous snapshots. This prepares us to derive the achievable capacity of the pipeline scheduling algorithm. The lower bound of the achievable continuous data collection capacity in a long-run is given in Theorem 2.4.3.

Theorem 2.4.3 *For a long-run continuous data collection, the lower bound of the achievable*

asymptotic network capacity of the pipeline scheduling algorithm is $\frac{nW}{12M\lceil(3.63\rho^2+c_3\rho+c_4)/H\rceil}$ *when* $\Delta_e \leq 12$ *or* $\frac{nW}{M\Delta_e\lceil(3.63\rho^2+c_3\rho+c_4)/H\rceil}$ *when* $\Delta_e > 12$, *where* $c_3 = \frac{8\pi}{\sqrt{3}}+\pi+2$ *and* $c_4 = \frac{8\pi}{\sqrt{3}}+2\pi+6$.

Proof: We prove Theorem 2.4.3 in two cases.

Case 1: $\Delta_e \leq 12$. In this case the number of the time slots used to collect N continuous snapshots is at most $M\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil(\Delta_e+15L'+12N-11)$ as proven in Theorem 2.4.2. Therefore, the lower bound of the capacity of the pipeline scheduling algorithm is

$$\frac{N\cdot nb}{tM\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil(\Delta_e+15L'+12N-11)} \tag{2.16}$$

$$= \frac{nb}{tM\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil(\frac{\Delta_e}{N}+\frac{15L'}{N}+12-\frac{11}{N})} \tag{2.17}$$

$$= \frac{nW}{M\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil(\frac{\Delta_e}{N}+\frac{15L'}{N}+12-\frac{11}{N})}. \tag{2.18}$$

When $N\to\infty$, the above equation approaches to $\frac{nW}{12M\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil}$. From Lemma 2.2.1, we have

$$2\beta_{\rho+2} \leq 2[\frac{\pi}{\sqrt{3}}(\rho+2)^2+(\frac{\pi}{2}+1)(\rho+2)+1] \tag{2.19}$$

$$= \frac{2\pi}{\sqrt{3}}\rho^2+(\frac{8\pi}{\sqrt{3}}+\pi+2)\rho+\frac{8\pi}{\sqrt{3}}+2\pi+6 \tag{2.20}$$

$$\approx 3.63\rho^2+c_3\rho+c_4, \tag{2.21}$$

where $c_3 = \frac{8\pi}{\sqrt{3}}+\pi+2$ and $c_4 = \frac{8\pi}{\sqrt{3}}+2\pi+6$. This implies the asymptotic network capacity in this case is $\frac{nW}{12M\lceil(3.63\rho^2+c_3\rho+c_4)/H\rceil}$.

Case 2: $\Delta_e > 12$. The lower bound of the asymptotic network capacity in this case is $\frac{nW}{M\Delta_e\lceil(3.63\rho^2+c_3\rho+c_4)/H\rceil}$, which can be proven similarly as in Case 1. □

In a dual-radio multi-channel WSN, since every node has two radios, the upper bound of the network capacity is $2W$. This is because the sink can receive at most two packets in one time slot. From Theorem 2.4.3, when $\Delta_e \leq 12$ and $M \leq \frac{n}{24\lceil(3.63\rho^2+c_3\rho+c_4)/H\rceil}$, or $\Delta_e > 12$ and $M \leq \frac{n}{2\Delta_e\lceil(3.63\rho^2+c_3\rho+c_4)/H\rceil}$, the achievable continuous data collection capacity of the pipeline scheduling algorithm is greater than $2W$. By checking the reasons carefully, we

find the pipeline scheduling and CDG are responsible for this improvement. By forming a CDG based pipeline, the time overlap of gathering multiple continuous snapshots conserves a lot of time, which accelerates the data collection process directly and significantly. These two reasons are also validated by the simulation results in Section 2.5.

Furthermore, we find that the pipeline scheduling algorithm is more effective for large scale WSNs, since large scale WSNs incur large size routing trees, which are more suitable for pipeline. The pipeline scheduling algorithm is also more effective for a long time continuous data collection, which can also be seen from Theorem 2.4.3.

For completeness, we also analyze the achievable network capacity of the pipeline scheduling algorithm in single-radio multi-channel WSNs. Now, since each sensor node has one radio, we make some modifications of the pipeline scheduling algorithm as follows. For the nodes in D_e, D_l $(1 \leq l \leq L')$, and C_l $(1 \leq l \leq L'-1)$, instead of transmitting the packets of the $(j+1)$-th $(j \geq 1)$ snapshot immediately after transmitting the packets of the j-th snapshot, they wait until their parent nodes have transmitted all the data packets of the j-th snapshot successfully (Note that the transmission of the data from the first snapshot does not have the waiting process.). For convenience, we refer to the modified pipeline scheduling algorithm as the *single-radio-based pipeline scheduling* algorithm. Then, by the similar proof technique shown in Theorem 2.4.1, the following lemma can be proven.

Lemma 2.4.3 *The number of the time slots used to collect the packets of the first snapshot by the single-radio-based pipeline scheduling algorithm for single-radio multi-channel WSNs is at most* $M \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil (\Delta_e + 15L' + 1)$.

On the basis of Lemma 2.4.3, the number of time slots used to collect all the packets of N continuous snapshots by the single-radio-based pipeline scheduling algorithm is shown in Theorem 2.4.4.

Theorem 2.4.4 *The time slots used by the single-radio-based pipeline scheduling algorithm to collect N continuous snapshots for single-radio multi-channel WSNs are at most*

$M\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil(\Delta_e+15L'+16N-15)$ *when* $\Delta_e\leq 12$ *or* $M\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil(N\Delta_e+15L'+4N-3)$ *when* $\Delta_e>12$.

Proof: Similar as the analysis in the proof of Theorem 2.4.2, when $\Delta_e\leq 12$, the sink will receive all the packets of a snapshot within every $12\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil+4\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil=16\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil$ STSs after it receives the packets of the first snapshot, which implies the number of time slots used to collect the packets of N continuous snapshots by the single-radio-based pipeline scheduling algorithm is at most

$$M\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil(\Delta_e+15L'+1)+(N-1)\cdot 16\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil \tag{2.22}$$

$$=M\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil(\Delta_e+15L'+16N-15). \tag{2.23}$$

Similarly, when when $\Delta_e>12$, the sink will receive all the packets of a snapshot within every $\Delta_e\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil+4\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil=(\Delta_e+4)\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil$ STSs after it receives the packets of the first snapshot, which implies the number of time slots used to collect all the packets of N continuous snapshots by the single-radio-based pipeline scheduling algorithm is at most

$$M\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil(\Delta_e+15L'+1)+(N-1)\cdot(\Delta_e+4)\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil \tag{2.24}$$

$$=M\left\lceil\frac{2\beta_{\rho+2}}{H}\right\rceil(N\Delta_e+15L'+4N-3). \tag{2.25}$$

□

Therefore, based on Theorem 2.4.4, the lower bound of the achievable continuous data collection capacity of the single-radio-based pipeline scheduling algorithm in a long-run is shown in Theorem 2.4.5.

Theorem 2.4.5 *For a long-run continuous data collection, the lower bound of the achievable asymptotic network capacity of the single-radio-based pipeline scheduling algorithm for single-radio multi-channel WSNs is* $\frac{nW}{16M\lceil(3.63\rho^2+c_3\rho+c_4)/H\rceil}$ *when* $\Delta_e\leq 12$ *or* $\frac{nW}{M(\Delta_e+4)\lceil(3.63\rho^2+c_3\rho+c_4)/H\rceil}$ *when* $\Delta_e>12$, *where* $c_3=\frac{8\pi}{\sqrt{3}}+\pi+2$ *and* $c_4=\frac{8\pi}{\sqrt{3}}+2\pi+6$.

Table 2.2 Comparison of the multi-path scheduling algorithm, the pipeline scheduling algorithm, and the best existing works (SDC = Snapshot Data Collection, CDC = Continuous Data Collection, IM = Interference Model, PrIM = Protocol Interference Model, PyIM = Physical Interference Model, RWN = Random Wireless Networks, AWN = Arbitrary Wireless Networks).

Algorithm name	SDC/CDC	IM	Υ
Zhu's algorithm [35]	SDC	PrIM	$\Theta(W)$
Chen's algorithm [4]	SDC	PrIM	$\Theta(W)$
Luo's algorithm (CDG) [1]	SDC	PrIM/PhIM	$\Theta(W)$
Chen's Algorithm [37]	SDC/CDC	PhIM	$\Omega(W)$
Multi-path scheduling	SDC	PrIM	$\Omega(\frac{W}{2\lceil(1.81\rho^2+c_1\rho+c_2)/H\rceil}) = \Omega(W)$
Pipeline scheduling	CDC	PrIM	$\Omega(\frac{nW}{12M\lceil(3.63\rho^2+c_3\rho+c_4)/H\rceil}) = \Omega(\frac{nW}{12M})$; or $\Omega(\frac{nW}{M\Delta_e\lceil(3.63\rho^2+c_3\rho+c_4)/H\rceil}) = \Omega(\frac{nW}{M\Delta_e})$;

Proof: By the similar technique in the proof of Theorem 2.4.3 and based on Theorem 2.4.4, this theorem holds. □

From Theorem 2.4.3 and Theorem 2.4.4, the capacity improvement ratio of the pipeline scheduling algorithm for multi-radio WSNs compared with the single-radio-based pipeline scheduling algorithm for single-radio WSNs is $\frac{4}{3}$ when $\Delta_e \leq 12$, or $\frac{\Delta_e+4}{\Delta_e}$ when $\Delta_e > 12$.

In summary, we compare the achievable network capacity of the proposed algorithms with the most recently published algorithms for data collection, and the result is shown in Table 2.2.

2.5 Simulations and Results Analysis

We conducted simulations to verify the performances of the proposed algorithms through implementing them with the C language. For all the simulations, we assume every WSN has one sink, and all the sensor nodes of each WSN are randomly distributed in a square area and the communication radius of each node is normalized to one. Suppose the network MAC layer works with TDMA, i.e. the network time can be slotted. Every node produces one data packet in a snapshot and the size of a packet is normalized to one. Every available channel

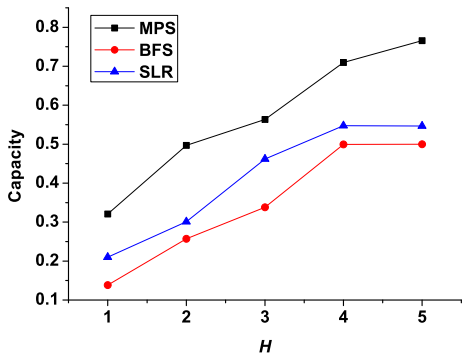
has the same bandwidth normalized to one. For any two different channels, we suppose they are orthogonal, i.e. the communications initialized over any two channels have no wireless interference. Furthermore, we assume a packet can be transmitted over a channel within a time slot.

The compared algorithms are BFS [4], SLR [54] and CDG [1]. BFS is a snapshot data collection algorithm based on a *breadth first search* tree and the scheduling is carried out path by path [4]. BFS is specifically proposed for single-radio single-channel WSNs. We extend it to the dual-radio multi-channel scenario in our simulations for fairness. SLR is a *straight-line routing* method for multi-unicast communication in multi-channel wireless networks with channel switching constraints [54]. For data collection, SLR works by setting every sensor having a unicast communication with the sink simultaneously. We also remove the channel switching constraints in SLR for fairness. Furthermore, we also implement the pipelined versions of BFS and SLR (i.e. add the pipeline technique to the data transmission in BFS and SLR), referred to as BFS-P and SLR-P respectively, when evaluate the performance of the proposed pipeline scheduling algorithm. The basic idea of CDG is discussed in Section 2.4. The proposed *multi-path scheduling* algorithm for snapshot data collection is referred to as MPS and the proposed *pipeline scheduling* algorithm for continuous data collection is referred to as PS in the following discussions.

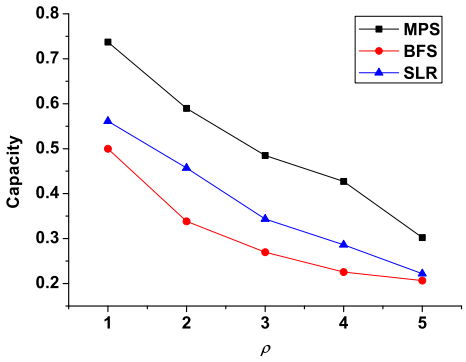
In the remainder of this section, we investigate the achievable capacities of MPS and PS through three groups of simulations respectively. In the simulations, H is the number of the available channels, ρ is the interference radius, n is the number of the sensors in a WSN, AR refers to the square area where a WSN is deployed, and N is the number of the snapshots in a continuous data collection task.

2.5.1 Performance of MPS

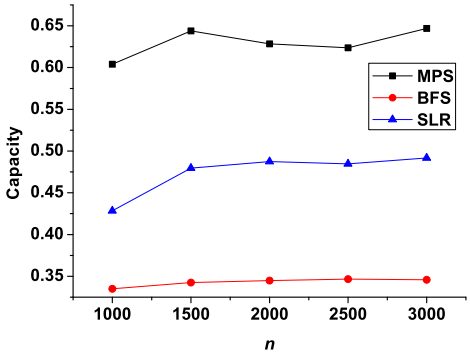
The snapshot data collection capacities of MPS, BFS, and SLR in different network scenarios are shown in Figure 2.7. In Figure 2.7(a), the capacity of every algorithm increases when the number of the available channels increases. This is because more available channels



(a) Capacity vs. H (ρ=2, n=4000, AR=30 $\times$ 30)



(b) Capacity vs. ρ (H=3, n=4000, AR=30 $\times$ 30)



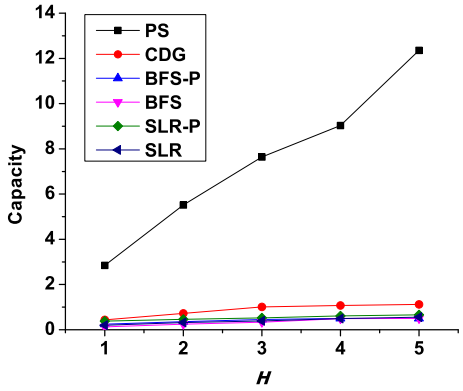
(c) Capacity vs. n (ρ=2, H=3, AR=20 $\times$ 20)

Figure 2.7 Snapshot data collection capacity (packets/time slot).

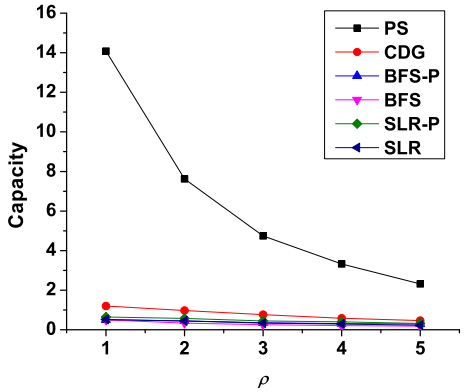
enable more concurrent transmissions, which accelerates the data collection process resulting in a higher capacity. After the number of the available channels arrives at 4, the capacities of BFS and SLR almost maintain the same level. This is because 4 channels are enough to prevent channel interference. However, radio confliction becomes the main barrier of a higher capacity at this time. MPS achieves a higher capacity compared with BFS and SLR. This is because MPS simultaneously schedules all the paths without radio confliction (except at the sink). Since radio confliction on a single path can be avoided easily, MPS can simultaneously schedule all the links without radio confliction on multiple paths, which implies MPS can make use of channels in a maximum degree. Whereas, BFS just schedules links without radio confliction on one path every time and SLR schedules all the transmission links simultaneously, which leads to serious radio confliction. On average, MPS achieves 77.49% and 41.95% more capacity than BFS and SLR, respectively.

The effect of the interference radius on the capacity is shown in Figure 2.7(b). With the increase of the interference radius, more transmission interference occurs, which leads to the decrease of the capacities of all the algorithms. Nevertheless, MPS still achieves the largest capacity since it simultaneously schedules multiple paths without radio confliction, which suggests a nice tradeoff between BFS and SLR. On average, MPS achieves 67.45% and 37.37% more capacity than BFS and SLR, respectively.

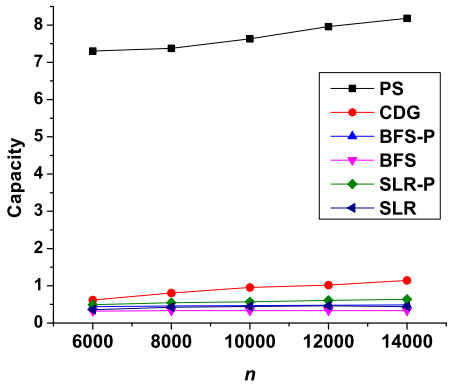
The effect of the number of the sensors on the capacity is shown in Figure 2.7(c). We can see that the number of the sensors in a network has a little impact on the capacities of MPS and SLR and almost no impact on the capacity of BFS. There are two reasons for this result. First, BFS is a single-path scheduling algorithm. Whatever the number of the sensors is, it schedules only one path every time. Second, the number of the channels is fixed to 2 in all of these three algorithms. This implies that whatever the number of the sensors is, they can simultaneously schedule at most two interfering links without radio confliction. On average, MPS achieves 83.51% and 32.87% more capacity than BFS and SLR, respectively.



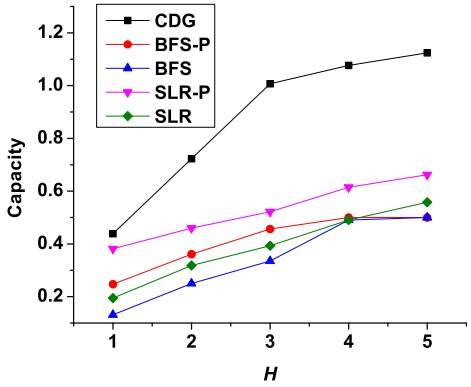
(a) Capacity vs. H (ρ=2, n=10000)



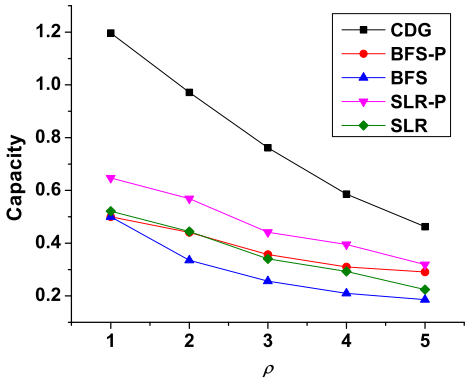
(b) Capacity vs. ρ (H=3, n=10000)



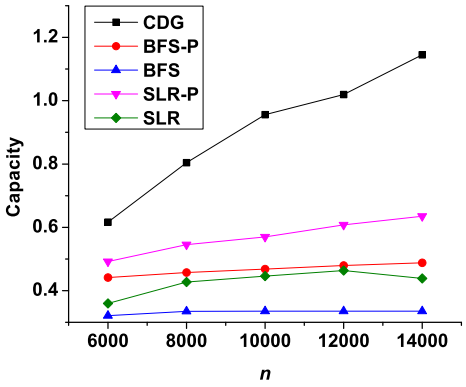
(c) Capacity vs. n (ρ=2, H=3)



(d) Capacity vs. H (ρ=2, n=10000)



(e) Capacity vs. ρ (H=3, n=10000)



(f) Capacity vs. n (ρ=2, H=3)

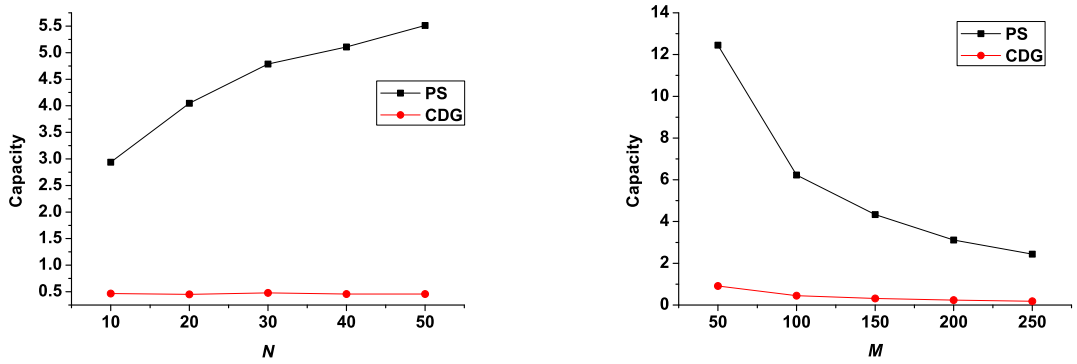
Figure 2.8 Continuous data collection capacity (packets/time slot) in different scenarios (AR=50 $\times$ 50, N=1000, M=100).

2.5.2 Performance of PS

The continuous data collection capacities of PS, CDG, BFS-P, BFS, SLR-P, and SLR in different network scenarios are shown in Figure 2.8. Figure 2.8(a) (respectively, Figure 2.8(b) and Figure 2.8(c)) and Figure 2.8(d) (respectively, Figure 2.8(e) and Figure 2.8(f)) are same except we do not show the achievable capacity of PS in Figure 2.8(d) (respectively, Figure 2.8(e) and Figure 2.8(f)). This is mainly for conveniently and clearly checking the achievable capacities of CDG, BFS-P, BFS, SLR-P, and SLR.

Figure 2.8(a) and (d) reflect the effect of the number of the available channels on the achievable continuous data collection capacity. As explained before, the capacities of all the algorithms increase as more and more channels are available. This is because more channels can prevent channel interferences among concurrent transmission links. PS has a much higher capacity compared with the other five algorithms. This is because: by forming a CDG based pipeline, the time overlap of gathering multiple continuous snapshots conserves a lot of time, which accelerates the data collection process directly and significantly in PS. Furthermore, PS collects data in the CDG way, which can reduce the overall data transmission times. This also explains why CDG has a higher capacity compared with BFS and SLR. From Figure 2.8(d), we can also see that BFS-P and SLR-R have higher network capacities than BFS and SLR, respectively. This is because the use of the pipeline technique can accelerate the data collection process. On average, PS achieves a capacity of 8.22 times of that of CDG, 17.1 times of that of BFS-P, 21.94 times of that of BFS, 13.17 times of that of SLR-P, and 18.39 times of that of SLR, respectively.

The effect of the interference radius on the capacity is shown in Figure 2.8(b) and (e). With the increase of the interference radius, a transmission link will interfere with more and more other transmission links, which leads to the decrease of capacity whatever algorithm it is. PS has the highest capacity among all the algorithms since it works with pipeline and CDG. On average, PS achieves a capacity of 7.31 times of that of CDG, 15.9 times of that of BFS-P, 20.43 times of that of BFS, 12.53 times of that of SLR-P, and 16.08 times of that of SLR, respectively.



(a) Impact of N (M=100) (b) Impact of M (N=100)

Figure 2.9 The impacts of N and M to the capacities (packets/time slot) of PS and CDG (ρ=2, H=3, n=5000, AR=50×50).

The effect of the number of the sensors on the capacity is shown in Figure 2.8(c) and (f). The increase of the number of the sensors has a little impact on BFS and SLR, and the reasons are similar to those explained in the previous subsection. Whereas, the capacities of PS and CDG have some improvement with more sensors in a WSN. This is because PS and CDG are more effective for large scale WSNs. On average, PS achieves a capacity of 8.77 times of that of CDG, 15.48 times of that of BFS-P, 23.15 times of that of BFS, 12.49 times of that of SLR-P, and 18.06 times of that of SLR, respectively.

2.5.3 Impacts of N and M

In this subsection, we investigate the impacts of the number of the snapshots and the value of M to the capacities of PS and CDG. As shown in Figure 2.9(a), with the increase of N in a continuous data collection task, PS achieves about 87.54% more capacity. This is straightforward from the analysis in Section 2.4. Since PS employs the pipeline technology, the transmissions of continuous snapshots are overlapped, which can significantly reduce the time used to collect all the snapshots data. With more snapshots in a continuous data collection task, the capacity of PS approaches closer and closer to its theoretical asymptotic capacity. For CDG, the number of the snapshots has little impact on its capacity.

Since the performance of CDG is depends on the value of M, the capacities of both PS and CDG decrease about 80% with the increase of the value of M as shown in Figure 2.9(b). This is because a bigger M implies more packets have to be transmitted for every sensor and longer transmission time for each snapshot is resulted. Nevertheless, considering that the value of M is usually much less than n, PS can still achieve a high capacity.

2.6 Conclusion

Motivated by the fact that there exist no works dedicated studying the capacity of continuous data collection for WSNs under the protocol interference model, we investigate this problem in dual-radio multi-channel WSNs in this part. We first propose a *multi-path scheduling* algorithm for the snapshot data collection problem and prove that its achievable network capacity is at least $\frac{W}{2\lceil(1.81\rho^2+c_1\rho+c_2)/H\rceil}$, where W is the channel bandwidth, H is the number of available orthogonal channels, ρ is the ratio of the interference radius over the transmission radius of a node, $c_1 = \frac{2\pi}{\sqrt{3}} + \frac{\pi}{2} + 1$, and $c_2 = \frac{\pi}{\sqrt{3}} + \frac{\pi}{2} + 2$. For the continuous data collection problem, we find that pipeline with the existing snapshot data collection methods cannot actually improve the network capacity. We explain the reason of this, and then propose a novel continuous data collection method for dual-radio multi-channel WSNs. This method speeds up the data collection process significantly. Theoretical analysis of this method shows that the achievable asymptotic network capacity is $\frac{nW}{12M\lceil(3.63\rho^2+c_3\rho+c_4)/H\rceil}$ when $\Delta_e \leq 12$ or $\frac{nW}{M\Delta_e\lceil(3.63\rho^2+c_3\rho+c_4)/H\rceil}$ when $\Delta_e > 12$, where n is the number of the sensors, M is a constant value and usually $M \ll n$, Δ_e is the maximum number of the leaf nodes having a same parent in the routing tree (i.e. data collection tree), $c_3 = \frac{8\pi}{\sqrt{3}} + \pi + 2$, and $c_4 = \frac{8\pi}{\sqrt{3}} + 2\pi + 6$. Furthermore, for completeness, we also analyze the performance of the proposed pipeline scheduling algorithm in single-radio multi-channel WSNs, which shows that for a long-run continuous data collection, the lower bound of the achievable asymptotic network capacity is $\frac{nW}{16M\lceil(3.63\rho^2+c_3\rho+c_4)/H\rceil}$ when $\Delta_e \leq 12$ or $\frac{nW}{M(\Delta_e+4)\lceil(3.63\rho^2+c_3\rho+c_4)/H\rceil}$ when $\Delta_e > 12$. Simulation results indicate that the proposed algorithms improve the network capacity significantly compared with existing works.

The future work of this part involves the following directions. First, in this work we study the snapshot data collection and continuous data collection problems for single/dual-radio multi-channel WSNs. We will extend this work to general multi-radio multi-channel WSNs to study the achievable capacities of snapshot data collection and continuous data collection. Second, the WSNs considered in this part are randomly deployed. We would like to further investigate the snapshot data collection/continuous data collection capacity in arbitrary WSNs. Third, the parameter M is crucial for the performance of the pipeline scheduling algorithm. Although the authors in [1] indicated that $M = 3K \sim 4K$ is usually sufficient for CDG, where K is a value determined by the correlations of the data of a snapshot and $K \ll n$, we also would like to derive the function relation between M and n for random WSNs, as well as arbitrary WSNs, in the future work. Finally, to study the snapshot data collection and continuous data collection capacities for arbitrary WSNs under the *general physical interference model* will be another future research direction.

PART 3

SNAPSHOT AND CONTINUOUS DATA COLLECTION IN PROBABILISTIC WIRELESS SENSOR NETWORKS

3.1 Introduction

As discussed in Part 1, after the seminal work [19], many works emerged to study the network capacity issue under the *protocol interference model* [4][5] or/and the *physical interference model* [13] for a variety of network scenarios, e.g. multicast capacity [12], unicast capacity [16], broadcast capacity [18][66], and snapshot data collection capacity [1][4]. All of the above mentioned works are based on the *deterministic network model*, where any pair of nodes in a network is either *connected* or *disconnected*. If two nodes are connected, i.e. there is a deterministic link between them, then a successful data transmission can be guaranteed as long as there is no collision. For the WSNs considered under the deterministic network model, we call them *deterministic WSNs*. However, in real applications, this deterministic network model assumption is too ideal and not practical due to the "*transitional region phenomenon*" [74][75]. With the transitional region phenomenon, a large number of network links (more than 90% [74]) become unreliable links, named *lossy links* [74]. Even without collisions, data transmission over a lossy link is successfully conducted with a certain probability, rather than being completely guaranteed. Therefore, a more practical network model for WSNs is the *probabilistic network model* [74], in which data communication over a link is successful with a certain probability rather than always successful or always failing. For convenience, the WSNs considered under the probabilistic network model are called *probabilistic WSNs*.

Recently, many efforts have been spent on the data collection issue. In [35][4][5][59][42][76], some tree-based data collection algorithms are proposed under the deterministic network model. In [35], the authors designed a family of path scheduling algorithms for SDC. Later on, the authors in [4][5][59] improved the path scheduling algorithms in [35] and implemented

order-optimal data collection methods with higher achievable capacity. Unlike [35][4][5][59], the authors in [42][76] studied the distributed data collection issue and designed an order-optimal distributed data collection algorithm. In [36][38][77][39][37], some data collection schemes are designed based on the cell-partition idea. Furthermore, by exploiting the geometrical properties of network distribution, the achievable data collection capacity are also analyzed in [36][38][77][39][37]. In [1], taking the advantage of the compressive data gathering technique, the authors in [1] designed a tree-based data collection algorithm. By analysis, they showed that the designed algorithm is order-optimal under both the PrIM and the PhIM. Unfortunately, for the data collection capacity issue, all the above mentioned existing works are based on the ideal deterministic network model rather than the more realistic probabilistic network model. Actually, lossy links may degrade the achievable network capacity of data collection since retransmissions may happen when transmit data, and thus more interference and congestion may be induced, followed by lower data transmission concurrency and efficiency. On the other hand, how these lossy links and retransmissions affect the snapshot and continuous data collection capacities is still an open problem. This motivates us to investigate the achievable network capacity of WSNs under the probabilistic network model.

Specifically, in this part, we study the achievable SDC and CDC capacity for probabilistic WSNs. Inspired by existing network partition methods [78], [34], we first investigate how to partition a probabilistic WSN into *cells* and *zones* to improve the concurrency of the data collection process. Subsequently, we propose two data collection schemes, the *Cell-based Path Scheduling* (CPS) algorithm and the *Zone-based Pipeline Scheduling* (ZPS) algorithm for SDC and CDC respectively. This work is dedicated to the data collection capacity issue for probabilistic WSNs and the main contributions are as follows:

1. For a probabilistic WSN deployed in a square area, we first partition the network into small *cells*. Then, we abstract each cell to a *super node* in the *data collection tree* built for data collection. Based on the data collection tree, we design a novel *Cell-based Path Scheduling* (CPS) algorithm for SDC. Theoretical analysis shows that the

achievable network capacity of CPS is $\Omega(\frac{1}{5\omega \ln n} \cdot W)$ in the sense of the worst case, and $\Omega(\frac{p_o}{2\omega} \cdot W)$ in the sense of expectation, where p_o is the *promising transmission threshold probability* defined in Section 3.2, ω is a constant defined in Section 3.3, and W is the data transmitting rate over a wireless channel, i.e. the channel bandwidth. Since the upper bound of the SDC capacity is shown to be W [4][5], CPS successfully achieves the order-optimal network capacity in the sense of expectation.

2. For the CDC problem in a probabilistic WSN, an intuitive idea is to employ a SDC method in a pipeline manner. However, this idea can only improve network capacity within a constant factor even in a deterministic WSN [5]. Therefore, by combining the Compressive Data Gathering (CDG) technique (a data gathering technique by exploiting the compressive sampling theory) [1] and the pipeline technique, we propose a novel *Zone-based Pipeline Scheduling* (ZPS) algorithm for CDC in probabilistic WSNs. Taking the benefits brought by CDG and pipeline, ZPS improves the achievable network capacity significantly. For collecting N continuous snapshots, we theoretically prove that the asymptotic achievable network capacity of ZPS is (a) $\Omega(\frac{N\sqrt{n}}{10\omega M\sqrt{\log n}\ln n} \cdot W)$ if $N = O(\sqrt{n/\log n})$ or $\Omega(\frac{n}{20\omega^2 M \log n \ln n} \cdot W)$ if $N = \Omega(\sqrt{n/\log n})$ in the sense of the worst case; and (b) $\Omega(\frac{p_o N\sqrt{n/\log n}}{4\omega M} \cdot W)$ if $N = O(\sqrt{n/\log n})$ or $\Omega(\frac{p_o n}{8\omega^2 M \log n} \cdot W)$ if $N = \Omega(\sqrt{n/\log n})$ in the sense of expectation, where n is the number of nodes in a WSN and M is a parameter used in CDG and usually $M \ll n$ in large-scale WSNs. Considering that the upper bound capacity is also W for CDC, this implies that the achievable network capacity of ZPS is $\frac{N\sqrt{n}}{\sqrt{\log n}\ln n}$ or $\frac{n}{\log n \ln n}$ times better than the optimal capacity of the snapshot data collection scenario in order in the sense of the worst case, and $\sqrt{n/\log n}$ or $n/\log n$ times better than the optimal capacity of the snapshot data collection scenario in order in the sense of expectation, which are significant improvements.

3. The simulation results also indicate that the proposed algorithms significantly improve the network capacity compared with the existing works for probabilistic and determin-

istic WSNs.

The rest of this part is organized as follows: Section 3.2 and Section 3.3 introduce the probabilistic network model and the network partition strategy which is crucial for the proposed data collection methods, respectively. The Cell-based Multi-Path Scheduling (CMPS) algorithm for snapshot data collection in probabilistic WSNs is proposed and analyzed in Section 3.4. Section 3.5 presents a novel Zone-based Pipeline Scheduling (ZPS) scheme for continuous data collection and its theoretical achievable asymptotic network capacity is shown. The simulations to validate the proposed algorithms are conducted in Section 3.6 and we conclude this part in Section 3.7.

3.2 Network Model

In this section, we describe the network model and assumptions. For the frequently used notations, we list them in Table 3.1 for convenience of referencing.

In this part, we consider a WSN consisting of n nodes, denoted by $s_1, s_2, \cdots, s_n$ respectively, and one sink (base station) deployed in a square plane with area $A = cn$ (i.e. the node density of the network is $1/c$), where c is a constant. Furthermore, we assume the distribution of all the nodes is i.i.d. (independent and identically distributed) and without loss of generality, the sink is located at the top-right corner of the square[1]. At each time interval, every node generates a data packet with size B bits, and transmits its data to the sink via a multi-hop way over a common wireless channel with bandwidth W bits/second, i.e. the data transmitting rate of the common channel is W. We further assume the time is slotted into time slots with each of length $t_o = B/W$ seconds.

During the data collection process, all the nodes in the network transmit data with an identical power P. Therefore, when node s_i transmits a packet to node s_j, the SINR

[1]Note that it is same with the situation when the sink is located at somewhere of the network, and we divide the network into four parts by a vertical line and a horizontal line, and consider each part individually.

Table 3.1 Notations in this part.

Parameter	Value
n	the number of sensor nodes
s_i	a sensor node
A	the size of the network deploying area
c, c_i, N_0	constants
η_i, μ_i^z	constants
B	the size of a data packet
W	the bandwidth of the wireless channel
t_o	a time slot
P	the working power of sensor nodes
$\Lambda(s_i, s_j)$	the SINR value at s_j associated with s_i
α	the path-loss exponent
$\Pr(s_i, s_j)$	the data transmission success probability from s_i to s_j
p_o	the promising transmission threshold probability
$\hbar$	the upper bound of retransmissions for a data packet over a lossy link
l	the length of a cell
m	the number of cells in a row/column
$\kappa_{i,j}$	a cell
$\chi_{i,j}$	the number of sensor nodes within cell $\kappa_{i,j}$
$\mathbb{S}_{i,j}$	the Compatible Transmission Cell Set (CTCS) containing cell $\kappa_{i,j}$
ω, r	constants, see Theorem 3.3.1
$o_{i,j}$	a compatible zone
$R = \omega l$	the length of a compatible zone
$s_{i,j}^u$	the super node corresponding to cell $\kappa_{i,j}$
$\mathbb{T}$	the data collection tree
S_i	a segment
L_j^i	the j-th level in segment S_i
t_p	the maximum super time slots consumed by a segment for transmitting one snapshot

(Signal-to-Interference-and-Noise-Ratio) associated with s_i at s_j is defined as

$$\Lambda(s_i, s_j) = \frac{P \cdot \|s_i - s_j\|^{-\alpha}}{N_0 + \sum_{s_k \in \mathcal{S}, s_k \neq s_i} P \cdot \|s_k - s_j\|^{-\alpha}}, \tag{3.1}$$

where $\|s_i - s_j\|$ is the Euclidean distance between s_i and s_j, α is the path-loss exponent and usually $\alpha \in (2, 4)$, $N_0 > 0$ is a constant representing the background noise, and $\mathcal{S}$ is the set of all the transmitters that transmit data simultaneously with s_i. Traditionally, in a *deterministic network model*, people assumed that if the SINR value at a node is greater than or equal to a threshold value, the packet can be received successfully. However, in real application environments, due to the existence of plenty of *lossy links*, this deterministic network model is too ideal. To be more practical and realistic, instead of taking the deterministic network model, we define a *probabilistic network model*, where each link is associated with a *success probability* which indicates the probability that a successful data transmission is conducted over this link. Based on the empirical literatures [75], we define the success probability associated with s_i and s_j as

$$\Pr(s_i, s_j) = (1 - \eta_1 \cdot e^{-\eta_2 \cdot \Lambda(s_i, s_j)})^{\eta_3}, \tag{3.2}$$

where η_1, η_2 and $\eta_3 > 1$ are positive constants. Clearly, when s_i transmits a data packet to s_j, until a successful transmission (i.e. s_j successfully received the whole data packet), the number of transmissions satisfies the *geometric distribution* with parameter $\Pr(s_i, s_j)$. Therefore, the expected transmission times from s_i to s_j is $1/\Pr(s_i, s_j)$, i.e. this transmission will cost $1/\Pr(s_i, s_j)$ time slots on average.

Actually, we do not want the success probability to be too low, which implies too many transmission times, too much energy consumption and induced interference, as well as low transmission concurrency. Therefore, we introduce a *promising transmission threshold probability* p_o here. For any promising transmission, we require its success probability is no less than the promising transmission threshold probability p_o, i.e. for any node pair s_i and

s_j, the transmission between s_i and s_j can be conducted only if $\Pr(s_i, s_j) \geq p_o$. Now, for any qualified communication pair to transmit one data packet, the expected transmission time is no more than t_o/p_o. For convenience, *in the sense of expectation*, we define a *modified time slot* $t_m = t_o/p_o$. Furthermore, we have Lemma **??** as follows, which indicates the upper bound of consumed time slots by any qualified communication to successfully transmit a data packet.

Lemma 3.2.1 *In a interference-free communication environment, it is almost sure that the number of consumed time slots of any qualified communication pair is upper bounded by*

$$\hbar = \arg \min_{1<z<1/(1-p_o)} 2\mu_1^z \ln n + \mu_2^z = O(\ln n),$$

where $\mu_1^z = -\frac{1}{\ln z(1-p_o)}$ *and* $\mu_2^z = -\log_{z(1-p_o)} \frac{p_o}{(1-p_o)(z-1)}$ *are some adjustable constant values depending on* z [2].

Proof: Please refer to the supplementary file. Suppose that the data transmission from s_i to s_j is a qualified communication, i.e. $\Pr(s_i, s_j) \geq p_o$. Let X be a random variable that denotes the number of consumed time slots for s_i to successfully transmit a data packet to s_j. Evidently, X is a *geometrically distributed random variable* with parameter $\Pr(s_i, s_j)$. Then, applying the Chernoff bound on X, we have

$$\Pr(X \geq \hbar) \tag{3.3}$$

$$\leq \min_{0<\xi<-\ln(1-p_o)} \frac{E[e^{\xi X}]}{e^{\xi \hbar}} \tag{3.4}$$

$$= \min_{0<\xi<-\ln(1-p_o)} \frac{p_o e^{-\xi\hbar}}{1-p_o}((1-(1-p_o)e^{\xi})^{-1} - 1). \tag{3.5}$$

[2] *Here, n is a notation that represents a large number. We exploit n to represent the upper bound of consumed time slots is mainly for the convenience of following derivations.*

Let $\xi = -\ln z(1-p_o)$, where $1 < z < 1/(1-p_o)$. Then, $e^{\xi} = \frac{1}{z(1-p_o)}$ and

$$\Pr(X \geq \hbar) \tag{3.6}$$

$$\leq \min_{1<z<1/(1-p_o)} \frac{p_o}{(1-p_o)(z-1)} e^{\ln z(1-p_o)\hbar} \tag{3.7}$$

$$= e^{-2\ln n} \tag{3.8}$$

$$= \frac{1}{n^2}. \tag{3.9}$$

$\sum_{n>0} \frac{1}{n^2} = \frac{\pi^2}{6}$ is a particular case of the Riemann Zeta function which is upper bounded. Thus, according to the Borel-Cantelli Lemma, $\Pr(X \leq \hbar) \sim 1$, i.e. it is almost sure that a qualified data communication can be successfully finished within $\hbar = \arg \min_{1<z<1/(1-p_o)} 2\mu_1^z \ln n + \mu_2^z$ time slots. □

Similarly, *in the sense of the worst case*, we define another *modified time slot* $t_w = \hbar \cdot t_o$ according to Lemma 3.2.1. In this part, we analyze the achievable snapshot and continuous data collection capacities in the sense of expectation and the worst case, respectively.

We further formally define the achievable *data collection capacity* as the ratio between the amount of data successfully collected by the sink and the time Γ used to collect these data. For instance, in our probabilistic WSN model, to collect N continuous snapshots, the achievable data collection capacity is defined as NnB/Γ, which is actually the data receiving rate at the sink. Particularly, when $N = 1$, nB/Γ is the SDC capacity.

3.3 Network Partition

In this section, we explain the network partition method, which is essential for our following data collection algorithm.

3.3.1 Cell-Based Network Partition

In the previous subsection, we assume the network is distributed over a square with area size $A = cn$. Now, we partition the network into small square *cells* with edge length $l = \sqrt{4c \log n}$ by a group of horizontal and vertical lines. The resulting network is shown

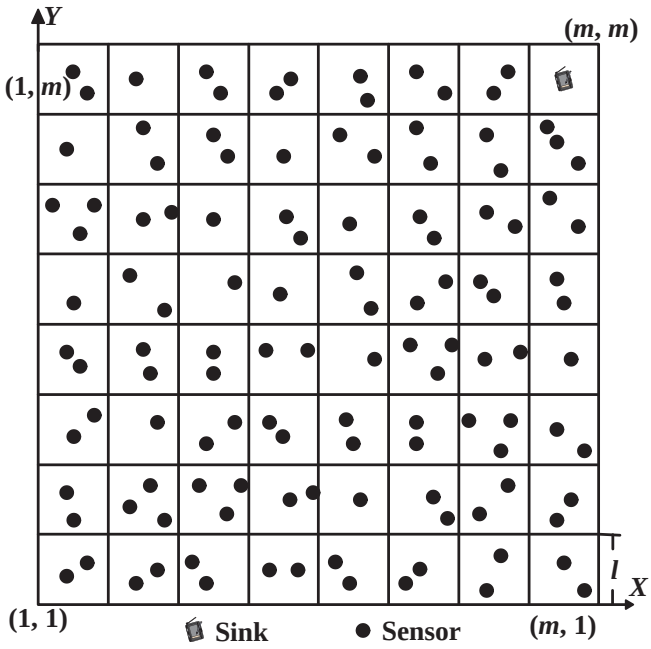


Figure 3.1 Network partition.

in Figure 3.1. For convenience, we use $m = \sqrt{cn}/\sqrt{4c\log n} = \sqrt{n/4\log n}$ to denote the number of cells in each column/row and further define $m' = m - 1$. For each cell shown in Figure 3.1, we assign each cell a pair of integer coordinates $(i, j)(1 \leq i, j \leq m)$, and a cell with coordinates (i, j) is denoted by $\kappa_{i,j}$. Clearly, the sink is located at the cell $\kappa_{m,m}$. Based on the network partition method, and considering that the sink is located at the top-right corner cell, we decide the possible communication modes for each cell (actually, for the nodes in each cell)[3] are *upward transmission*, *rightward transmission*, and *up-rightward transmission*. Take cell $\kappa_{i,j}$ as an example, when $\kappa_{i,j}$ works on the upward (respectively, rightward, up-rightward) transmission mode, it transmits its data to cell $\kappa_{i,j+1}$ (respectively, $\kappa_{i+1,j}$, $\kappa_{i+1,j+1}$). For cell $\kappa_{i,j}(1 \leq i, j \leq m)$, let the random variable $\chi_{i,j}$ denote the number of nodes within it. Then, based on the above network partition, the following three lemmas can be derived.

Lemma 3.3.1 *The expected number of nodes $E[\chi_{i,j}]$, i.e. the average number of nodes, in*

[3]Without confusion, we use cell and the nodes within this cell interchangeably.

$\kappa_{i,j}(1 \leq i, j \leq m)$ *is* $4 \log n$.

Proof: Since all the nodes are i.i.d., for any node, it is located at $\kappa_{i,j}$ with probability $l^2/A = 4 \log n/n$. Hence, the number of nodes within $\kappa_{i,j}$ is a *binomial random variable* with parameters $(n, 4 \log n/n)$. Thus, $E[\chi_{i,j}] = n \cdot 4 \log n/n = 4 \log n$. □

Lemma 3.3.2 *It is almost surely that no cell is empty, i.e. it is almost surely that* Pr(*there exists at least one cell with no nodes*) $\cong 0$ *for large* n.

Proof: For any cell $\kappa_{i,j}$, let $e_{i,j}$ to denote the event that $\kappa_{i,j}$ is empty, i.e. $\chi_{i,j} = 0$. As explained in Lemma 3.3.1, $\chi_{i,j}$ satisfies the binomial distribution with parameters $(n, 4 \log n/n)$. Then, applying the Chernoff bound and for any $\xi < 0$, we have

$$\Pr(\chi_{i,j} = 0) \leq \Pr(\chi_{i,j} \leq 1) \leq \min_{\xi<0} \Pr(e^{\xi \chi_{i,j}} \geq e^{\xi}) \tag{3.10}$$

$$\leq \min_{\xi<0} \frac{E[e^{\xi \chi_{i,j}}]}{e^{\xi}} \tag{3.11}$$

$$= \min_{\xi<0} \frac{[1 + (e^{\xi} - 1) \cdot 4 \log n/n]^n}{e^{\xi}} \tag{3.12}$$

$$\leq \min_{\xi<0} \frac{e^{4 \log n(e^{\xi}-1)}}{e^{\xi}} \quad (by\ 1 + x \leq e^x) \tag{3.13}$$

$$= \min_{\xi<0} \exp(4(e^{\xi} - 1) \log n - \xi) \tag{3.14}$$

$$\leq \exp(-3 \log n) \quad (for\ large\ n) \tag{3.15}$$

$$\leq \frac{1}{n^3}. \tag{3.16}$$

Then, according to Boole's inequality, we have the probability of the event that there exists at least one cell with no nodes as follows.

$$\Pr(\text{there exists at least one cell with no nodes}) \tag{3.17}$$

$$= \Pr(\bigcup_{1\le i,j\le m} \chi_{i,j} = 0) \tag{3.18}$$

$$\le \sum_{1\le i,j\le m} \frac{1}{n^3} \tag{3.19}$$

$$= \frac{n}{4\log n} \cdot \frac{1}{n^3} \tag{3.20}$$

$$= \frac{1}{4n^2 \log n}. \tag{3.21}$$

Based on the Borel-Cantelli Lemma, we conclude that it is almost surely that no cell is empty for large n. □

Lemma 3.3.3 *It is almost surely that no cell contains more than* $10\log n$ *nodes.*

Proof: For any cell $\kappa_{i,j}$, applying the Chernoff bound and for any $\xi > 0$, we have

$$\Pr(\chi_{i,j} > 10\log n) \le \min_{\xi>0} \frac{E[e^{\xi\chi_{i,j}}]}{e^{10\xi\log n}} \tag{3.22}$$

$$= \min_{\xi>0} \frac{[1+(e^{\xi}-1)4\log n/n]^n}{e^{10\xi\log n}} \tag{3.23}$$

$$\le \min_{\xi>0} \exp(4(e^{\xi}-1)\log n - 10\xi\log n) \tag{3.24}$$

$$= \min_{\xi>0} \exp((4e^{\xi}-4-10\xi)\log n) \tag{3.25}$$

$$\le \exp(-3\log n) \tag{3.26}$$

$$\le \frac{1}{n^3}. \tag{3.27}$$

Similarly, according to Boole's inequality, we obtain the probability that there exists at least one cell contains more than $10\log n$ nodes as follows.

$$\Pr(\bigcup_{1\leq i,j\leq m}\chi_{i,j} > 10\log n) \leq \sum_{1\leq i,j\leq m}\frac{1}{n^3} \tag{3.28}$$

$$= \frac{1}{4n^2\log n}. \tag{3.29}$$

Again, based on the Borel-Cantelli Lemma, we conclude that it is almost surely that no cell contains more than $10\log n$ nodes for large n. □

From Lemma 3.3.1 we know that the expected number of nodes within a cell is $4\log n$. Lemma 3.3.2 implies that for large WSN, i.e. large n, every cell will have some nodes within it. Furthermore, from Lemma 3.3.3, the probability that a cell contains more than $10\log n$ is zero when $n \to \infty$. Hence, in the following discussion, we assume a cell contains $4\log n$ nodes *in the sense of expectation* and $10\log n$ nodes *in the sense of the worst case*.

3.3.2 Zone-Based Network Partition

After partitioning the network into cells, we want to find which cells can carry out transmissions concurrently. Further, for these cells that can conduct transmissions concurrently, we define them as a *Compatible Transmission Cell Set* (CTCS), denoted by $\mathbb{S}$. Formally, we define $\mathbb{S} = \{\kappa_{i1,j1}, \kappa_{i2,j2}, \cdots, \kappa_{ig,jg} |$ (1) $1 \leq ik, jk \leq m$ for $1 \leq k \leq g$; (2) $\kappa_{ik,jk}(1 \leq k \leq g)$ can conduct transmissions concurrently; (3) For $\kappa_{ik,jk}(1 \leq k \leq g)$, suppose $\kappa'_{ik,jk}$ is its destination, i.e. $\kappa_{ik,jk}$ transmits data to $\kappa'_{ik,jk}$, then when $\kappa_{ik,jk}(1 \leq k \leq g)$ conduct transmissions simultaneously, $\min\limits_{1\leq k\leq g}\Pr(\kappa_{ik,jk}, \kappa'_{ik,jk}) = \min\limits_{1\leq k\leq g}\min\{\Pr(s_u, s'_u)|s_u$ is a node in $\kappa_{ik,jk}$, and s'_u is a node/sink in $\kappa'_{ik,jk}\} \geq p_o.\}$. Clearly, the CTCS is an *equivalence relation* defined on the cells (i.e. CTCS is reflexive, symmetric, and transitive). Hence, a CTCS can be viewed as an *equivalence class*.

In order to partition the cells of a WSN into equivalence classes, i.e. CTCSs, we assign each cell $\kappa_{i,j}(1 \leq i, j \leq m)$ a vector representation $\vec{\kappa_{i,j}} = ((i-1)\cdot l, (j-1)\cdot l) = \kappa_{i,j}$. We further introduce two vectors $\vec{X} = (R, 0)$ and $\vec{Y} = (0, R)$, where $R = \omega \cdot l$, $\omega \in \mathbb{Z}$.

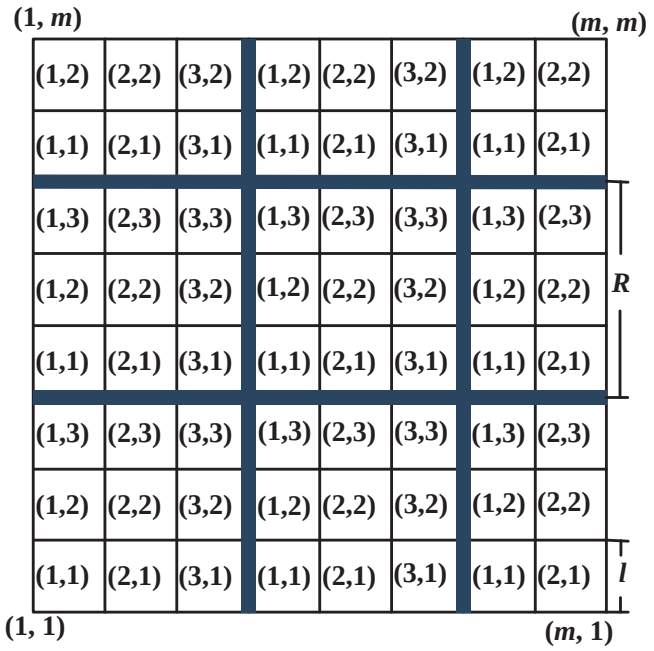


Figure 3.2 Equivalence classes (CTCS) and zones.

Then, for any cell $\kappa_{i,j}(1 \leq i,j \leq m)$, we define the equivalence class, i.e. the CTCS, containing $\kappa_{i,j}$ as the set $\mathbb{S}_{i,j} = \{\kappa_{i,j}^{\vec{}} + a \cdot \vec{X} + b \cdot \vec{Y} | a,\ b \in \mathbb{Z}\}$, i.e. $\mathbb{S}_{i,j} = \{\kappa_{i+a\cdot\omega,j+b\cdot\omega} | a,\ b \in \mathbb{Z}, 1 \leq i + a \cdot \omega, j + b \cdot \omega \leq m\}$ (Here, we suppose $\mathbb{S}_{i,j}$ is a CTCS. Later we will show how to choose R to make it actually a CTCS.). Taking the WSN shown in Figure 3.1 as an example, if we set $\omega = 3$, i.e. $R = 3l$, then the network can be partitioned into 9 equivalence classes, i.e. CTCSs, $\mathbb{S}_{i,j}(1 \leq i,j \leq 3)$ as shown in Figure 3.2. In Figure 3.2, the CTCS containing $\kappa_{1,1}$ is $\mathbb{S}_{1,1} = \{\kappa_{1,1}, \kappa_{4,1}, \kappa_{7,1}, \kappa_{1,4}, \kappa_{4,4}, \kappa_{7,4}, \kappa_{1,7}, \kappa_{4,7}, \kappa_{7,7}\}$. Now, we start from the bottom-left corner of the WSN and partition the network into square zones, named *compatible zones*, with edge length $R = \omega \cdot l$ as shown in Figure 3.2 (where $\omega = 3$). Similar to denote a cell, for each compatible zone, we use $o_{i,j}(1 \leq i,j \leq \lceil m/\omega \rceil)$ to denote it, and the bottom-left zone with the smallest i and j, i.e. $o_{1,1}$. Clearly, $o_{i,j} = \{\kappa_{i',j'} | (i-1) \cdot \omega + 1 \leq i' \leq i \cdot \omega, (j-1) \cdot \omega + 1 \leq j' \leq j \cdot \omega\}$, i.e. within a compatible zone, none of the cells belong to the same equivalence class. Furthermore, all the cells with the same relative position in different compatible zones belong to the same equivalence, i.e. the same CTCS.

Now, to make any $\vec{X}, \vec{Y}$-based cell set $\mathbb{S}_{i,j}$ actually a CTCS, we need to decide the value

of R. For large WSNs, the value of R is determined by the following Theorem 3.3.1.

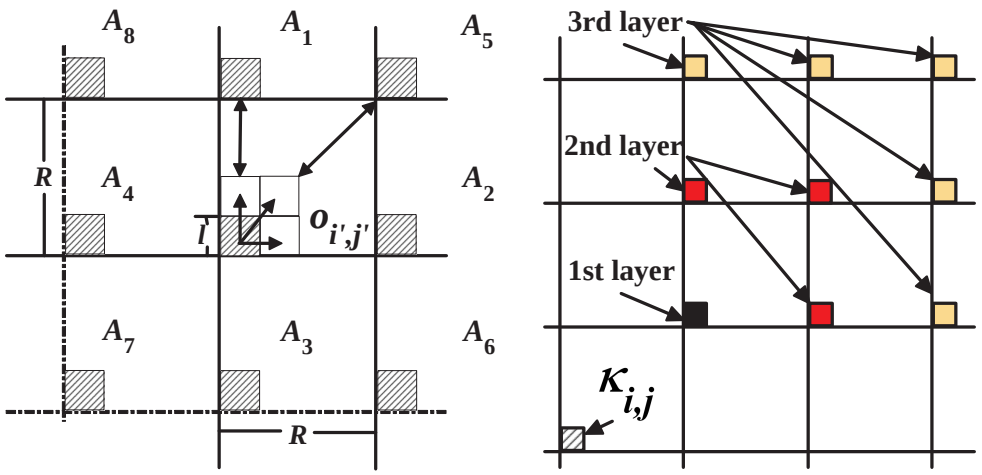
Theorem 3.3.1 *Let* $R = \omega \cdot l$, $\omega = \Theta(\frac{r+o(1)}{l})$, $r = 2\sqrt{2}l$ [4], $\vec{X} = (R, 0)$, $\vec{Y} = (0, R)$, *then the set* $\mathbb{S}_{i,j} = \{\vec{\kappa_{i,j}} + a \cdot \vec{X} + b \cdot \vec{Y} | a, b \in \mathbb{Z}\} = \{\kappa_{i+a\cdot\omega, j+b\cdot\omega} | a, b \in \mathbb{Z}, 1 \leq i + a \cdot \omega, j + b \cdot \omega \leq m\}$ *is a CTCS.*

Before proving Theorem 3.3.1, we prove Lemma 3.3.4 and Lemma 3.3.5 first. In the following proof, assume all the cells in a CTCS $\mathbb{S}_{i,j}$ conduct transmissions concurrently, and all other cells keep quiet or receive data from some cells in $\mathbb{S}_{i,j}$.

Lemma 3.3.4 *For each* $\mathbb{S}_{i,j}$, $\forall \kappa_{i,j} \in \mathbb{S}_{i,j}$, *suppose* $\kappa'_{i,j}$ *is the destination cell of* $\kappa_{i,j}$, *then* $\Lambda(\kappa_{i,j}, \kappa'_{i,j}) = \min\{\Lambda(s_u, s_v) | 1 \leq u, v \leq n, s_u \in \kappa_{i,j}, s_v \in \kappa'_{i,j}\} \geq \frac{P \cdot r^{-\alpha}}{N_0 + P \cdot \beta \cdot R^{-\alpha}}$, *where* $r = 2\sqrt{2}l$ *and* β *is a positive constant.*

Proof : For an arbitrary cell $\kappa_{i,j} \in \mathbb{S}_{i,j}$, suppose it is located in zone $o_{i',j'}$ as shown in Figure 3.3(a). Let $\vec{h} = \vec{\kappa_{i,j}}$, then the compatible cells of $\kappa_{i,j}$ in $\mathbb{S}_{i,j}$ can be partitioned into eight disjointed subsets, denoted by A_k $(1 \leq k \leq 8)$, as shown in Figure 3.3(a), where each A_k is defined as follows:

[4] *r is a parameter in the derivation, which can be viewed as the maximum transmission range of a node.*



(a) (b)

Figure 3.3 (a) Interference areas and (b) cell layered of A_5.

$$
\begin{cases}
A_1 \equiv \{\vec{h} + b \cdot \vec{Y} | b \in \mathbb{Z}^+\} \equiv \{\kappa_{i,j+b\cdot\omega} | b \in \mathbb{Z}^+\} \\
A_2 \equiv \{\vec{h} + a \cdot \vec{X} | a \in \mathbb{Z}^+\} \equiv \{\kappa_{i+a\cdot\omega,j} | a \in \mathbb{Z}^+\} \\
A_3 \equiv \{\vec{h} + b \cdot \vec{Y} | b \in \mathbb{Z}^-\} \equiv \{\kappa_{i,j+b\cdot\omega} | b \in \mathbb{Z}^-\} \\
A_4 \equiv \{\vec{h} + a \cdot \vec{X} | a \in \mathbb{Z}^-\} \equiv \{\kappa_{i+a\cdot\omega,j} | a \in \mathbb{Z}^-\} \\
A_5 \equiv \{\vec{h} + a \cdot \vec{X} + b \cdot \vec{Y} | a, b \in \mathbb{Z}^+\} \\
\quad \equiv \{\kappa_{i+a\cdot\omega,j+b\cdot\omega} | a, b \in \mathbb{Z}^+\} \\
A_6 \equiv \{\vec{h} + a \cdot \vec{X} + b \cdot \vec{Y} | a \in \mathbb{Z}^+, b \in \mathbb{Z}^-\} \\
\quad \equiv \{\kappa_{i+a\cdot\omega,j+b\cdot\omega} | a \in \mathbb{Z}^+, b \in \mathbb{Z}^-\} \\
A_7 \equiv \{\vec{h} + a \cdot \vec{X} + b \cdot \vec{Y} | a, b \in \mathbb{Z}^-\} \\
\quad \equiv \{\kappa_{i+a\cdot\omega,j+b\cdot\omega} | a, b \in \mathbb{Z}^-\} \\
A_8 \equiv \{\vec{h} + a \cdot \vec{X} + b \cdot \vec{Y} | a \in \mathbb{Z}^-, b \in \mathbb{Z}^+\} \\
\quad \equiv \{\kappa_{i+a\cdot\omega,j+b\cdot\omega} | a \in \mathbb{Z}^-, b \in \mathbb{Z}^+\}
\end{cases}
. \tag{3.30}
$$

Then, for any node (sender) s_u in $\kappa_{i,j}$ and its corresponding receiver s_v under any communication mode, we consider the achievable $\Lambda(s_u, s_v)$. Evidently, $\|s_u - s_v\| \leq r$ under any communication mode. Furthermore, since P is fixed (P is fixed for every node, and

meanwhile P has to be large enough to guarantee the communications of neighboring cells) and N_0 is a constant, the value of $\Lambda(s_u, s_v)$ depends on $\sum_{w \neq u} \|s_w - s_v\|^{-\alpha}$ only. Considering that $\mathbb{S}_{i,j} \setminus \{\kappa_{i,j}\}$ has been partitioned into $A_k (1 \leq k \leq 8)$, we consider $s_w \in A_k$ $(1 \leq k \leq 8)$ separately. In the following derivation, we use the facts that $R \geq 3l$, and $\alpha \in (2, 4)$.

Case 1: $s_w \in A_1$. In this case, we have $\|s_w - s_v\| \geq b \cdot R - 2l$, which implies

$$\sum_{s_w \in A_1} \|s_w - s_v\|^{-\alpha} \tag{3.31}$$

$$\leq \sum_{b \geq 1} (b \cdot R - 2l)^{-\alpha} \tag{3.32}$$

$$= R^{-\alpha} \cdot [(1 - \frac{2l}{R})^{-\alpha} + \sum_{b>1} (b - \frac{2l}{R})^{-\alpha}] \tag{3.33}$$

$$\leq R^{-\alpha} \cdot [(1 - \frac{2}{3})^{-\alpha} + \sum_{b>1} (b-1)^{-2}] \tag{3.34}$$

$$= R^{-\alpha} \cdot ((\frac{1}{3})^{-\alpha} + \frac{\pi^2}{6}) \tag{3.35}$$

$$= c_1 \cdot R^{-\alpha}, \tag{3.36}$$

where $c_1 = (\frac{1}{3})^{-\alpha} + \frac{\pi^2}{6}$. Similarly, we can prove that for Case g $(2 \leq g \leq 4, s_w \in A_g)$, $\sum_{s_w \in A_g} \|s_w - s_v\|^{-\alpha} \leq c_g \cdot R^{-\alpha}$, where $c_g (2 \leq g \leq 4)$ are some positive constants.

Case 2: $s_w \in A_5$. In this case, the cells in A_5 can be layered with respect to $\kappa_{i,j}$ with the δ-th layer having $2\delta - 1$ cells[5] as shown in Figure 3.3(b). Furthermore, the distance between s_v and any node s_w in the δ-th layer is greater than $\delta R - 2l$, i.e. $\|s_w - s_v\| \geq \delta R - 2l$ for s_w

[5]This can be proven by *mathematical induction*.

located in the cell at the δ-th layer. Hence

$$\sum_{s_w \in A_5} \|s_w - s_v\|^{-\alpha} \tag{3.37}$$

$$\leq \sum_{\delta \geq 1} (2\delta - 1)(\delta R - 2l)^{-\alpha} \tag{3.38}$$

$$= R^{-\alpha} \sum_{\delta \geq 1} (2\delta - 1)(\delta - \frac{2l}{R})^{-\alpha} \tag{3.39}$$

$$\leq R^{-\alpha} \sum_{\delta \geq 1} (2\delta - 1)(\delta - \frac{2}{3})^{-\alpha} \tag{3.40}$$

$$\leq R^{-\alpha} \cdot [3^{\alpha} + \sum_{\delta \geq 2} (2\delta - 1)(\delta - 1)^{-\alpha}] \tag{3.41}$$

$$= R^{-\alpha} \cdot [3^{\alpha} + \sum_{\hat{\delta} \geq 1} (2\hat{\delta} + 1)\hat{\delta}^{-\alpha}] \tag{3.42}$$

$$= R^{-\alpha} \cdot [3^{\alpha} + \sum_{\hat{\delta} \geq 1} (2\hat{\delta}^{1-\alpha} + \hat{\delta}^{-\alpha})] \tag{3.43}$$

$$= R^{-\alpha} \cdot (3^{\alpha} + 2\zeta(\alpha - 1) + \zeta(\alpha)) = c_5 R^{-\alpha}, \tag{3.44}$$

where $\zeta(\cdot)$ is the *Riemann zeta function* and $\zeta(\alpha - 1) \leq \frac{1}{\alpha - 2}$, $\zeta(\alpha) \leq \frac{\pi^2}{6}$. In the derivation, $\hat{\delta} = \delta - 1$.

Similarly, we can prove that for Case g ($6 \leq g \leq 8, s_w \in A_g$), $\sum_{s_w \in A_g} \|s_w - s_v\|^{-\alpha} \leq c_g \cdot R^{-\alpha}$, where $c_g (6 \leq g \leq 8)$ is a positive constant. In summary,

$$\sum_{w \neq u} \|s_w - s_v\|^{-\alpha} = \sum_{g=1}^{8} \sum_{s_w \in A_g} \|s_w - s_v\|^{-\alpha} \tag{3.45}$$

$$\leq \sum_{g=1}^{8} c_g \cdot R^{-\alpha}. \tag{3.46}$$

Let $\beta = \sum_{g=1}^{8} c_g$, we have $\Lambda(s_u, s_v) \geq \frac{P \cdot r^{-\alpha}}{N_0 + P \cdot \beta \cdot R^{-\alpha}}$. Since $s_u \in \kappa_{i,j}$ and $s_v \in \kappa'_{i,j}$ are arbitrarily chosen, this lemma holds. □

Lemma 3.3.5 *$\mathbb{S}_{i,j}$ is a CTCS when $R \geq (c_9 \cdot r^{-\alpha} + c_{10})^{-1/\alpha}$, where $c_9 = \frac{\eta_2}{\beta \cdot \ln \eta_1 (1 - \eta \sqrt[3]{p_o})^{-1}}$ and $c_{10} = -\frac{N_0}{P \cdot \beta}$.*

Proof: To prove this lemma, we need to show $\forall \kappa_{i,j} \in \mathbb{S}_{i,j}$, suppose $\kappa'_{i,j}$ is the destination cell of $\kappa_{i,j}$, $\Pr(\kappa_{i,j}, \kappa'_{i,j}) = \min\{\Pr(s_u, s_v) | s_u \in \kappa_{i,j}, \text{ and } s_v \in \kappa'_{i,j}\} \geq p_o$ when $R \geq (c_9 \cdot r^{-\alpha} + c_{10})^{-1/\alpha}$. This is equivalent to show $\forall s_u \in \kappa_{i,j}, \forall s_v \in \kappa'_{i,j}$, $\Pr(s_u, s_v) = (1 - \eta_1 \cdot e^{-\eta_2 \cdot \Lambda(s_u, s_v)})^{\eta_3} \geq p_o$. Hence, it is sufficiency to have

$$(1 - \eta_1 \cdot \exp(-\eta_2 \cdot \frac{P \cdot r^{-\alpha}}{N_0 + P \cdot \beta \cdot R^{-\alpha}}))^{\eta_3} \geq p_o \tag{3.47}$$

$$\Leftrightarrow \exp(-\eta_2 \cdot \frac{P \cdot r^{-\alpha}}{N_0 + P \cdot \beta \cdot R^{-\alpha}}) \leq \frac{1 - \sqrt[\eta_3]{p_o}}{\eta_1} \tag{3.48}$$

$$\Leftrightarrow \eta_2 \cdot \frac{P \cdot r^{-\alpha}}{N_0 + P \cdot \beta \cdot R^{-\alpha}} \geq \ln \frac{\eta_1}{1 - \sqrt[\eta_3]{p_o}} \tag{3.49}$$

$$\Leftrightarrow R \geq (\frac{\eta_2}{\beta \cdot \ln \eta_1 (1 - \sqrt[\eta_3]{p_o})^{-1}} \cdot r^{-\alpha} - \frac{N_0}{P \cdot \beta})^{-1/\alpha}. \tag{3.50}$$

Let $c_9 = \frac{\eta_2}{\beta \cdot \ln \eta_1 (1 - \sqrt[\eta_3]{p_o})^{-1}}$ and $c_{10} = -\frac{N_0}{P \cdot \beta}$, the conclusion holds. □

Now, we are ready to prove Theorem 3.3.1.

Proof of Theorem 3.3.1: From Lemma 3.3.5, we know that when $R \geq (c_9 \cdot r^{-\alpha} + c_{10})^{-1/\alpha}$, $\mathbb{S}_{i,j}$ is a CTCS. Since large $|\mathbb{S}_{i,j}|$ implies more concurrent data transmissions, we prefer small R. Thus, let $R = (c_9 \cdot r^{-\alpha} + c_{10})^{-1/\alpha}$. Define $\omega = \lceil R/l \rceil$. For large n, i.e. large-scale WSNs, $R \sim \Theta(r + o(1))$, which implies $\omega = \Theta(\frac{r + o(1)}{l})$. Thus, the conclusion of Theorem 3.3.1 holds. □

From Theorem 3.3.1, we know that if we set $R = \omega \cdot l$, then, all the CTCSs can conduct data transmissions simultaneously in an interference-free manner. Based on the conclusion of Theorem 3.3.1, the following corollary can be obtained.

Corollary 3.3.1 *By $\vec{X}$ and $\vec{Y}$, the cells $\kappa_{i,j} (1 \leq i, j \leq m)$ can be partitioned into at most ω^2 CTCSs (equivalence classes).*

Proof: From Theorem 3.3.1, we know that $\mathbb{S}_{i,j} = \{\vec{\kappa_{i,j}} + a \cdot \vec{X} + b \cdot \vec{Y} | a, b \in \mathbb{Z}\} = \{\kappa_{i + a \cdot \omega, j + b \cdot \omega} | a, b \in \mathbb{Z}, 1 \leq i + a \cdot \omega, j + b \cdot \omega \leq m\}$. Therefore, for each cell $\kappa_{i',j'} (1 \leq i', j' \leq m)$, $\kappa_{i',j'} \in \mathbb{S}_{i,j}$ if $(i' \mod \omega) = (i \mod \omega)$ and $(j' \mod \omega) = (j \mod \omega)$. Since both $(i' \mod \omega)$ and $(j' \mod \omega)$ have ω distinct values for all $1 \leq i', j' \leq n$, we have at most ω^2 CTCSs, i.e. equivalence classes. □

3.4 Snapshot Data Collection

In this section, we study the achievable network capacity of SDC. First, we propose a novel Cell-based Path Scheduling (CPS) algorithm for SDC. Subsequently, we analyze the achievable network capacity of CPS. Finally, we make some further discussion about the extension from SDC to CDC.

3.4.1 Cell-based Path Scheduling (CPS)

Before giving the CPS algorithm, we construct a *data collection tree*, which serves as the routing structure, for the data collection process. For each cell $\kappa_{i,j}(1 \leq i, j \leq m)$, we abstract it to a *super node*, denoted by $s^u_{i,j}$[6]. Following the discussion in Section 3.3.1, a cell contains $4\log n$ nodes in the sense of expectation and $10\log n$ nodes in the sense of the worst case. Thus, we abstract the data packets of nodes within a cell as a *super data packet*, whose size is $4\log n \cdot B$ bits in the sense of expectation and $10\log n \cdot B$ bits in the sense of the worst case. Accordingly, to send out a super data packet, we define a *super time slot* t_s as $4\log n \cdot t_m = 4t_o \log n/p_o$ *in the sense of expectation* and $10\log n \cdot t_w = 10\hbar \log n \cdot t_o$ *in the sense of the worst case*. Afterwards, considering the communication modes defined in Section 3.3.1, we construct a data collection tree, denoted by $\mathbb{T}$, rooted at the sink to connect all the super nodes according to the following rules: 1) For super nodes $s^u_{i,j}(1 \leq i, j \leq m')$ (note that $m' = m - 1$), $s^u_{i,j}$ transmits its data to $s^u_{i+1,j+1}$, i.e. create a link from $s^u_{i,j}$ to $s^u_{i+1,j+1}$. 2) For super nodes $s^u_{m,j}(1 \leq j \leq m')$, $s^u_{m,j}$ transmits its data to $s^u_{m,j+1}$, i.e. create a link from $s^u_{m,j}$ to $s^u_{m,j+1}$. 3) For super nodes $s^u_{i,m}(1 \leq i \leq m')$, $s^u_{i,m}$ transmits its data to $s^u_{i+1,m}$, i.e. create a link from $s^u_{i,m}$ to $s^u_{i+1,m}$. After applying the above rules to all the super nodes except for $s^u_{m,m}$, the data collection tree is built. Taking the WSN shown in Figure 3.1 as an example, the obtained data collection tree is shown in Figure 3.4. For a data transmission route from a leaf super node to the root in $\mathbb{T}$, we call it a *path*. The path starting from $s^u_{i,1}(1 \leq i \leq m)$ is denoted by P_i and the path from $s^u_{1,j}(2 \leq j \leq m)$ is denoted

[6]Without confusion, we use cell and super node exchangeably.

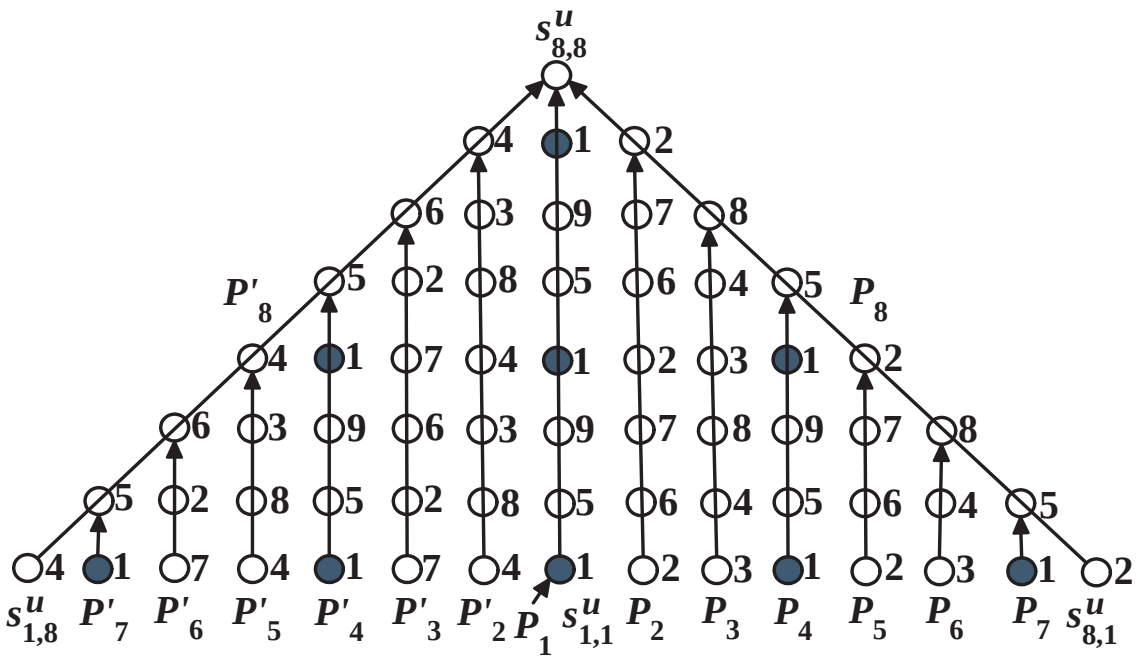


Figure 3.4 Data collection tree.

by P'_j, as shown in Figure 3.4.

According to Corollary 3.3.1, all the cells of a WSN can be partitioned into ω^2 CTCSs (equivalence classes). For each CTCS $\mathbb{S}_{i,j}(1 \leq i, j \leq \omega)$, we map it to an integer $(i-1)\cdot\omega+j$. In Figure 3.4, the number next to each super node indicates the CTCS it belongs to. For convenience, we also use $\mathbb{S}_{(i-1)\cdot\omega+j}$ to represent the CTCS $\mathbb{S}_{i,j}(1 \leq i, j \leq \omega)$.

Based on the abstracted data collection tree $\mathbb{T}$, we propose a novel *Cell-based Path Scheduling* (CPS) algorithm, which has two phases. In Phase I of CPS, we schedule the ω^2 CTCSs one by one, until all the data packets of cells $\kappa_{i,j}(1 \leq i, j \leq m')$ have been collected to the cells on path P_m, path P'_m, or the sink. In Phase II of CPS, we schedule the cells of P_m and P'_m until all the data packets have been collected to the sink. We use the example shown in Figure 3.4 to present the main idea of CPS as follows. The formal description of CPS is shown in Algorithm 2[7].

Phase I: Inner-Tree Scheduling. Since the cells within a CTCS can be scheduled

[7]Note that, although the two phases are not shown explicitly in Algorithm 2, the data collection process can be viewed consisting of two phases as discussed.

Algorithm 2: The CPS Algorithm

input : a data collection tree, CTCSs
output: a data collection plan

```
while the sink has not collected all the data do
    for i = 1; i ≤ ω; i++ do
        for j = 1; j ≤ ω; j++ do
            if all the cells in CTCS 𝕊_{i,j} have no data for transmission then
                continue;
            Assign CTCS 𝕊_{i,j} a super time slot;
            During the assigned super time slot, schedule all the super nodes (cells)
            in 𝕊_{i,j} simultaneously: for ∀κ_{u,v} ∈ 𝕊_{i,j}, schedule all the nodes with data
            for transmission in κ_{u,v} sequentially according to some order, e.g. the ID
            order, each with one modified time slot;
```

to transmit data concurrently, schedule CTCSs $\mathbb{S}_1, \mathbb{S}_1, \cdots, \mathbb{S}_{\omega^2}$ orderly, each for a super time slot. Repeat Phase I until there is no packet remaining at the super node $s^u_{i,j}(1 \leq i, j \leq m')$, i.e. all the data packets at $s^u_{i,j}(1 \leq i, j \leq m')$ have been collected to the sink or $s^u_{i,j}(i = m$ or $j = m)$. For the specific nodes within a cell, schedule them sequentially according to any order at the available super time slots for this cell[8]. Taking the data collection tree $\mathbb{T}$ shown in Figure 3.4 as an example, the cells in $\mathbb{T}$ can be partitioned into 9 CTCSs. For the 9 CTCSs $\mathbb{S}_1, \mathbb{S}_1, \cdots, \mathbb{S}_9$, we schedule them orderly each for one super time slot. At the end of Phase I, all the data packets of $s^u_{i,j}(1 \leq i, j \leq 7)$ have been collected to the sink, or the cells on path P_8 and P'_8.

Phase II: Scheduling of P_m and P'_m. For the super nodes $s^u_{i,j}(i = m$ or $j = m)$ which have data packets waiting for collection, partition them into λ CTCSs (Actually, $\lambda \leq 2\omega - 1$

[8]Suppose the parent node of super node $s^u_{i,j}$ is $s^u_{i',j'}$, i.e. all the nodes in cell $\kappa_{i,j}$ will transmit their data to the nodes in cell $\kappa_{i',j'}$. Then, when a node s_u in cell $\kappa_{i,j}$ is scheduled to transmit data to some node in cell $\kappa_{i',j'}$, s_u will transmit its data to the node s_v in cell $\kappa_{i',j'}$, where s_v satisfies the condition that the success probability of the link from s_u to s_v is the highest among the links from s_u to all the nodes in cell $\kappa_{i',j'}$.

Now, assume the success probability of the link from s_u to s_v is 0.5. Then, when s_u transmits a data packet to s_v, s_v successfully receives this data packet with probability 0.5. If this data transmission fails, s_u will retransmit that data packet until the packet is successfully received by s_v. Evidently, the expected transmission times of that packet is 2 in this case.

In this part, without specification, for any node s_u, it determines its next hop and transmits data in terms of the aforementioned manner.

which is proven in Lemma 3.4.4.). Then, schedule these λ CTCSs sequentially each for one super time slot. Repeat Phase II until all the packets have been collected to the sink. Taking P_8 and P_8' shown in Figure 3.4 as an example, the cells on P_8 and P_8' can be partitioned into 5 CTCSs. Then, we schedule these 5 CTCSs sequentially until all the data packets been collected to the sink.

From the description of CPS, we know it can collect all the data packets to the sink after Phase I and Phase II. In the following subsection, we will analyze the achievable network capacity of CPS.

3.4.2 Capacity Analysis of CPS

In this subsection, we investigate the achievable network capacity of CPS. The upper bound of SDC is W even under the deterministic network model [4][5] [9]. Therefore, the upper bound of SDC under the probabilistic network model is W too. Consequently, we focus on the lower bound of CPS in the following analysis.

For convenience, we introduce the concept of *scheduling round*. A scheduling round for Phase I (respectively, Phase II) of CPS is the time used to run Phase I (respectively, Phase II) once. For the data collection tree $\mathbb{T}$ shown in Figure 3.4, a scheduling round is $9t_s$ (respectively, $5t_s$) in Phase I (respectively, Phase II), since there are 9 (respectively, 5) CTCSs need to schedule in each running of Phase I (respectively, Phase II). Now, we can obtain the number of super time slots used in Phase I of CPS as shown in Lemma 3.4.1.

Lemma 3.4.1 *For SDC, it takes CPS $\omega^2 m'$ super time slots to finish Phase I.*

Proof: According to the scheduling in Phase I, every CTCS is scheduled once in a scheduling round. This implies every super node in the network is scheduled once in every scheduling round. Therefore, for each super node $s^u_{i,j} (1 \leq i, j \leq m')$, it can receive one super data packet at most from its child and send out one super data packet at most to its parent

[9] This is because the sink node can receive at most one data packet during a time slot. Consequently, based on the definition of data collection capacity (which is defined as the average data receiving rate of the sink during a data collection process), W is a trivial upper bound of any data collection algorithm in both deterministic WSNs and probabilistic WSNs.

during every scheduling round. Thus, for each path of $P_i(1 \leq i \leq m')$ and $P'_j(2 \leq j \leq m')$, its length will decrease by one after each scheduling round (if we assume the node without any data for transmission will be deleted from the path). It follows that the data packets of $s^u_{i,j}(1 \leq i, j \leq m')$ will be collected to the sink or $s^u_{i,j}(i = m \text{ or } j = m)$ in m' scheduling round, i.e. $\omega^2 m'$ super time slots, since the length of the longest path of $P_i(1 \leq i \leq m')$ and $P'_j(2 \leq j \leq m')$ is m'. □

Now, we study the time slots used in Phase II of CPS. First, we derive the number super data packets remaining at each of the super nodes $s^u_{i,j}(i = m \text{ or } j = m)$ waiting for transmission at the beginning of Phase II. Subsequently, we obtain the upper bound of the number of super time slots used in Phase II, and followed by the lower bound of the achievable network capacity of CPS. In the following analysis, we use $\phi_{i,j}(1 \leq i, j \leq m)$ to denote the number of super data packets transmitted/forwarded by $s^u_{i,j}$ through the entire SDC process. Further, we use $\varphi_{i,j}(1 \leq i, j \leq m)$ to denote the number of super data packets at $s^u_{i,j}$ waiting for transmission at the beginning of Phase II. Clearly, $\varphi_{i,j} = 0(1 \leq i, j \leq m')$ after Phase I.

Lemma 3.4.2 *For* $1 \leq i \leq m'$, $\phi_{m,i} = \frac{i(i+1)}{2}$.

Proof: Based on the constructed data collection tree in the previous subsection, for $s^u_{m,i}(2 \leq i \leq m')$, it has two children $s^u_{m-1,i-1}$ and $s^u_{m,i-1}$. Hence, during the entire data collection process, the number of super data packets transmitted/forwarded by $s^u_{m,i}(2 \leq i \leq m')$ is the sum of the number of super data packets transmitted/forwarded by $s^u_{m-1,i-1}$ and $s^u_{m,i-1}$ plus 1 (1 means the super data packet of $s^u_{m,i}$ itself), i.e. $\phi_{m,i} = \phi_{m-1,i-1} + \phi_{m,i-1} + 1$. Considering $\phi_{m-1,i-1}$, it has only one child $\phi_{m-2,i-2}$. Thus, $\phi_{m-1,i-1} = \phi_{m-2,i-2} + 1$. In a sum, we have

$$\begin{cases} \phi_{m,1} = 1, \ \phi_{m-i+1,1} = 1 \\ \phi_{m-1,i-1} = \phi_{m-2,i-2} + 1 \\ \phi_{m,i} = \phi_{m-1,i-1} + \phi_{m,i-1} + 1 \end{cases} \tag{3.51}$$

Then, it is straightforward for us to obtain the generating functions of $\phi_{m-1,i-1}$ which is $\phi_{m-1,i-1} = i-1$, and $\phi_{m,i}(1 \leq i \leq m')$ which is $\phi_{m,i} = \frac{i(i+1)}{2}$. □

From the proof of Lemma 3.4.2 and by symmetry, we have the following corollary.

Corollary 3.4.1 *For* $1 \leq i \leq m'$, $\phi_{i,m} = \frac{i(i+1)}{2}$.

Based on Lemma 3.4.2, we obtain the number of super data packets at $s^u_{m,i}$ waiting for transmission at the beginning of Phase II as shown in Lemma 3.4.3.

Lemma 3.4.3 *Let* $\theta = \lceil \frac{\sqrt{1+8m'}-1}{2} \rceil$, *then*

$$\varphi_{m,i} = \begin{cases} 0, & 1 \leq i < \theta \\ \phi_{m,i} - m' = \frac{i(i+1)}{2} - m' \leq i, & i = \theta \\ i, & \theta < i \leq m' \end{cases} \tag{3.52}$$

Proof: We prove this lemma by cases.

Case 1: $1 \leq i < \theta$. From Lemma 3.4.2, $s^u_{m,i}$ transmits/forwards $\phi_{m,i} = \frac{i(i+1)}{2}$ super data packets to its parent through the entire SDC process. In Phase I, we schedule every CTCS for m' times by the proof of Lemma 3.4.1, which implies $s^u_{m,i}$ has been scheduled for m' times. It follows that $s^u_{m,i}$ can transmit/forward m' super data packets to its parent during its available super time slots in Phase I. Considering that $1 \leq i < \theta$, we have $\phi_{m,i} = \frac{i(i+1)}{2} \leq \frac{1}{2}(\theta^2 + \theta) = \frac{1}{2} \cdot 2m' = m'$. Thus, we conclude that $s^u_{m,i}(1 \leq i < \theta)$ has already finished its data transmission task in Phase I, i.e. $\varphi_{m,i}(1 \leq i < \theta) = 0$ at the beginning of Phase II.

Case 2: $i = \theta$. According to the proof of the previous case and the scheduling of Phase I, for super node $s^u_{m,\theta}$, its two children $s^u_{m,i-1}$ and $s^u_{m-1,i-1}$ have no data packet waiting for transmission at the beginning of Phase II. Furthermore, as explained in the previous case, $s^u_{m,\theta}$ has been scheduled for m' times in Phase I, which implies that $s^u_{m,i}$ transmitted m' super data packets to its parent. It follows that the number of data packets waiting at $s^u_{m,\theta}$ for transmission is $\varphi_{m,i} = \phi_{m,i} - m' = \frac{i(i+1)}{2} - m' \leq i$ at the beginning of Phase II.

Case 3: $\theta < i \leq m'$. For the child $s^u_{m-1,i-1}$ of $s^u_{m,i}$, it transmitted $i-1$ super data packets to $s^u_{m,i}$ in Phase I by the proof of Lemma 3.4.2. For another child $s^u_{m,i-1}$ of $s^u_{m,i}$, it transmitted m' super data packets to $s^u_{m,i}$ in Phase I by the proof of Lemma 3.4.1. Furthermore, $s^u_{m,i}$ also transmitted m' super data packets to its parent $s^u_{m,i+1}$ by the proof of Lemma 3.4.1. This implies the number of super data packets waiting at $s^u_{m,i}(\theta < i \leq m')$ for transmission at the beginning of Phase II is $\varphi_{m,i} = (i-1)+1 = i$. □

According to Corollary 3.4.1 and Lemma 3.4.3, it is straightforward to obtain the following corollary.

Corollary 3.4.2

$$\varphi_{i,m} = \begin{cases} 0, & 1 \leq i < \theta \\ \phi_{m,i} - m' = \frac{i(i+1)}{2} - m' \leq i, & i = \theta \\ i, & \theta < i \leq m' \end{cases} \tag{3.53}$$

Lemma 3.4.4 *For super nodes $s^u_{m,i}(\theta \leq i \leq m')$ and $s^u_{j,m}(\theta \leq j \leq m')$, they can be partitioned into at most $2\omega - 1$ CTCSs, i.e. $\lambda \leq 2\omega - 1$, where λ is the one in Phase II of CPS.*

Proof: According to the vector-based CTCS partition method in Section 3.3.2, the super nodes $s^u_{m,i}(\theta \leq i \leq m)$ can be partitioned into at most ω CTCSs. Similarly, $s^u_{j,m}(\theta \leq j \leq m)$ can be partitioned into at most ω CTCSs too. Furthermore, $s^u_{m,m}$ lies in the same CTCS no matter how to partition these cells, which implies $s^u_{m,i}(\theta \leq i \leq m')$ and $s^u_{j,m}(\theta \leq j \leq m')$ can be partitioned into at most $2\omega - 1$ CTCSs. □

Lemma 3.4.5 *In Phase II of the CPS algorithm, it costs at most $\frac{1}{2}(2\omega-1)(m'+\theta)(m'-\theta+1)$ super time slots to transmit all the data packets to the sink.*

Proof: During each schedule round of Phase II, every super node of $s^u_{m,i}(\theta \leq i \leq m')$ and $s^u_{j,m}(\theta \leq j \leq m')$ is scheduled once to transmit a super data packet to its parent. Hence, the sink will receive two super data packets during every scheduling round. From Lemma

3.4.3 and Corollary 3.4.2, we know that the total number of super data packets waiting at $s^u_{m,i}(\theta \le i \le m')$ and $s^u_{j,m}(\theta \le j \le m')$ for transmission at the beginning of Phase II is at most $2\sum_{i=\theta}^{m'} = (m'+\theta)(m'-\theta+1)$. It turns out that the sink can collect all the super data packets at $s^u_{m,i}$ and $s^u_{j,m}$ within $\frac{1}{2}(2\omega-1)(m'+\theta)(m'-\theta+1)$. □

Now, we are ready to derive the achievable network capacity of CPS in the sense of the worst case and in the sense of expectation as shown in Theorem 3.4.1.

Theorem 3.4.1 *For the achievable data collection capacity of CPS for SDC, it is* $\Omega(\frac{1}{5\omega \ln n} \cdot W)$ *in sense of the worst case, which is a degradation of* $O(\ln n)$ *of the optimum capacity, and* $\Omega(\frac{p_o}{2\omega} \cdot W)$ *in the sense of expectation, which is order-optimal.*

Proof: From Lemma 3.4.1 and Lemma 3.4.5, the total number of super time slots used by CPS is at most

$$\omega^2 m' + \frac{1}{2}(2\omega-1)(m'+\theta)(m'-\theta+1) \tag{3.54}$$
$$\le \omega^2 m + \frac{1}{2}\cdot 2\omega(m+\theta)(m-\theta) \tag{3.55}$$
$$\le \omega^2 m + \omega m^2 \tag{3.56}$$
$$= \omega^2\sqrt{\frac{n}{2\log n}} + \frac{\omega n}{2\log n} \tag{3.57}$$
$$\le O(\frac{\omega n}{2\log n}). \tag{3.58}$$

The total amount of data received by the sink is $n \cdot b$. Thus, in the sense of the worst case, the achievable network capacity of CPS is

$$\frac{n\cdot B}{O(\frac{\omega n}{2\log n})\cdot t_s} = \frac{n\cdot B}{O(\frac{\omega n}{2\log n})\cdot 10\log n\cdot t_w} \tag{3.59}$$
$$= \frac{n\cdot B}{O(\frac{\omega n}{2\log n})\cdot 10\log n\cdot \hbar t_o} \tag{3.60}$$
$$= \Omega(\frac{1}{5\omega\hbar}\cdot W) \tag{3.61}$$
$$= \Omega(\frac{1}{5\omega\ln n}\cdot W). \tag{3.62}$$

Similarly, in the sense of expectation, the achievable network capacity of CPS is

$$\frac{n \cdot B}{O(\frac{\omega n}{2 \log n}) \cdot t_s} = \frac{n \cdot B}{O(\frac{\omega n}{2 \log n}) \cdot 4 \log n \cdot \frac{t_o}{p_o}} \tag{3.63}$$

$$= \Omega(\frac{p_o}{2\omega} \cdot W). \tag{3.64}$$

Since the upper bound of SDC is W under deterministic/probabilistic network model, and p_o, ω are constants, the achievable network capacity of CPS in the sense of expectation is order-optimal. However, the data collection capacity of CPS has a degradation of $O(\ln n)$ in the sense of the worst case. □

When addressing the CDC problem, an intuitive idea is to combine the existing SDC methods with the *pipeline* technique. Nevertheless, such an idea cannot induce a significant improvement on the network capacity. Taking the CPS as an example, it has already achieved the order-optimal data collection capacity. By pipelining the CPS algorithm, data transmissions at the nodes far from the sink can definitely be accelerated. However, the fact that the sink can receive at most one packet during each time slot makes the data accumulated at the nodes near the sink. As a result, the network capacity still cannot be improved even with pipeline [5].

3.5 Continuous Data Collection

Intuitively, CDC has much more traffic load than SDC. Therefore, it is easier for the data to accumulate at the nodes near the sink, which makes the data transmission schedule very complicated and inefficient. Consequently, new elegant techniques are required to address this situation. On the other hand, the combination of a SDC method and the *pipeline* technique cannot improve network capacity effectively. Therefore, we propose a novel *Zone-based Pipeline Scheduling* (ZPS) algorithm based on the technology used in *Compressive Data Gathering* (CDG) [1] in this section. The basic idea of CDG is discussed in Section 2.4. Theoretical analysis shows that ZPS can improve the data collection capacity significantly.

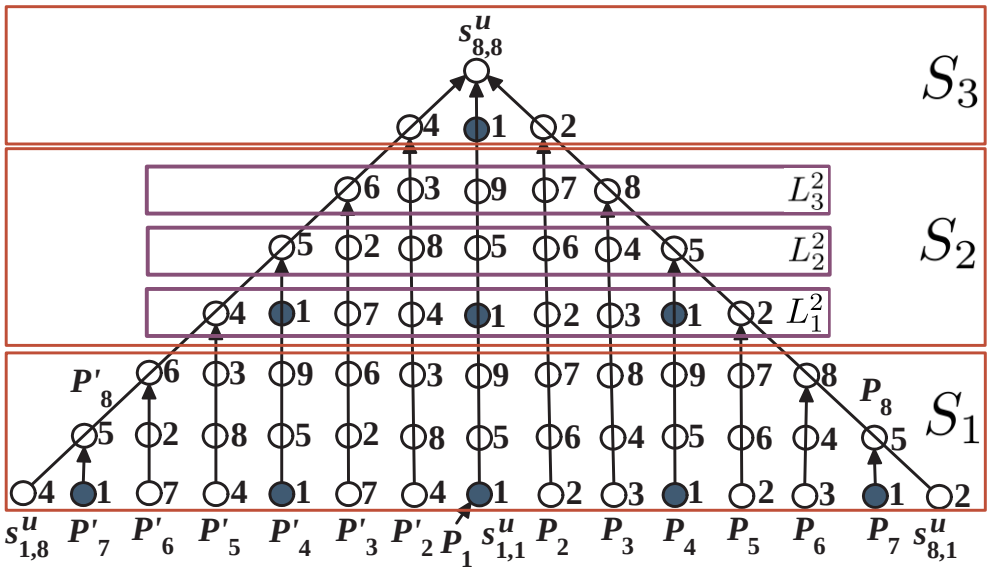


Figure 3.5 Levels and segments.

3.5.1 Zone-based Pipeline Scheduling

Considering the benefit brought by CDG, we combine it with the pipeline technique to design an efficient CDC algorithm, named the *Zone-based Pipeline Scheduling* (ZPS) algorithm. Before giving the detailed design of ZPS, we further partition the data collection tree $\mathbb{T}$ constructed in Section 3.4.1 into *levels* and *segments*, which are sets of cells (super nodes) and compatible zones, respectively. As shown in Section 3.3.2, a WSN can be partitioned into $(\lceil m/\omega \rceil)^2$ compatible zones. For these zones, we define the set $\{o_{j,i}, o_{i,j} | i \leq j \leq \lceil m/\omega \rceil\}(1 \leq i \leq \lceil m/\omega \rceil)$ as a *segment*, denoted by $S_i(1 \leq i \leq \lceil m/\omega \rceil)$. Within segment $S_i(1 \leq i \leq \lceil m/\omega \rceil)$, we define the set $\{s^u_{y,x}, s^u_{x,y} | x = (i-1) \cdot \omega + j, x \leq y \leq m\}(1 \leq j \leq \omega)$ as a *level*, denoted by $L^i_j(1 \leq j \leq \omega)$. Taking the $\mathbb{T}$ shown in Figure 3.4 as an example, it can be partitioned into 3 segments as shown in Figure 3.5, where $S_1 = \{o_{1,1}, o_{2,1}, o_{3,1}, o_{1,2}, o_{1,3}\}$, $S_2 = \{o_{2,2}, o_{3,2}, o_{2,3}\}$, and $S_3 = \{o_{3,3}\}$. Within a segment, the super nodes can be partitioned into ω levels, e.g. in Figure 3.5, within S_2, the super nodes can be partitioned into levels $L^2_1 = \{s^u_{4,4}, s^u_{5,4}, s^u_{6,4}, s^u_{7,4}, s^u_{8,4}, s^u_{4,5}, s^u_{4,6}, s^u_{4,7}, s^u_{4,8}\}$, $L^2_2 = \{s^u_{5,5}, s^u_{6,5}, s^u_{7,5}, s^u_{8,5}, s^u_{5,6}, s^u_{5,7}, s^u_{5,8}\}$, and $L^2_3 = \{s^u_{6,6}, s^u_{7,6}, s^u_{8,6}, s^u_{6,7}, s^u_{6,8}\}$.

Based on the definitions of segment, level and CTCS, we observe that (i) for level-

s $L_j^i(1 \leq i \leq \lceil m/\omega \rceil)$, all their super nodes (cells), i.e. $\bigcup_{i=1}^{\lceil m/\omega \rceil} L_j^i$, come from CTCSs $\mathbb{S}_{j,k} \bigcup \mathbb{S}_{j,k}(j \leq k \leq \omega)$, i.e. the super nodes in $\bigcup_{i=1}^{\lceil m/\omega \rceil} L_j^i$ can be partitioned into at most $2\omega - 1$ CTCSs; and (ii) on the other hand, for every super node (cell) in $\mathbb{S}_{j,k} \bigcup \mathbb{S}_{j,k}(j \leq k \leq \omega)$, it is located at level L_j^i for some $1 \leq i \leq \lceil m/\omega \rceil$. According to the observations, we design a *Zone-based Pipeline Scheduling* (ZPS) algorithm for CDC, which consists of *inter-segment pipeline scheduling* and *intra-segment scheduling* as follows.

Inter-Segment Pipeline Scheduling. Since the super nodes in levels $L_j^i(1 \leq i \leq \lceil m/\omega \rceil)$ can be partitioned into $2\omega - 1$ CTCSs, we can take each level as an unit and schedule the $j - th(1 \leq j \leq \omega)$ level of all the segments $S_i(1 \leq i \leq \lceil m/\omega \rceil)$ simultaneously. In other words, we can schedule all the segments concurrently as long as we schedule the same $j - th(1 \leq j \leq \omega)$ level within each segment. Therefore, when we collect N continuous snapshots, we can pipeline the data transmission on the segments, i.e. for each segment $S_i(1 \leq i \leq \lceil m/\omega \rceil)$, S_i starts to transmit the data packets of the $(k+1)$-th $(k > 0)$ snapshot immediately after it transmits all the data of the k-th snapshot to segment S_{i+1}. Suppose $t(S_i)(1 \leq i \leq \lceil m/\omega \rceil)$ is the number of super time slots used by segment S_i to transmit all the data packets of a snapshot to the subsequent segment (or the sink) and let $t_p = \max\{t(S_i) | 1 \leq i \leq \lceil m/\omega \rceil\}$. Then, a segment data transmission pipeline on all the segments is formed with each segment works with t_p super time slots for every snapshot (Now, a snapshot is equivalent to an individual task in a traditional pipeline operation). By this data transmission pipeline, the sink can receive the data of a snapshot in every t_p super time slots after it receives the data of the first snapshot.

Intra-Segment Scheduling. The inter-segment pipeline scheduling provides a scheme to form a data transmission pipeline over all the segments. Clearly, the efficiency of the formed pipeline highly depends on t_p, which is determined by the intra-segment scheduling. Within segment $S_i(1 \leq i \leq \lceil m/\omega \rceil)$ to transmit the k-th snapshot, we schedule the super nodes level by level, i.e. schedule $L_1^i, L_2^i, \cdots, L_\omega^i$ sequentially to transmit the k-th snapshot. Finally, the data packets of the k-th snapshot are transmitted to the next segment by the super nodes in level L_ω^i. When schedule $L_j^i(1 \leq j \leq \omega)$ for the k-th snapshot, we first

partition the super nodes in L_j^i into at most $2\omega - 1$ CTCSs according to the observations. Subsequently, we schedule these $2\omega - 1$ CTCSs sequentially. When schedule a particular CTCS, we let all the super nodes within this CTCS transmit their data in the CDG way, i.e. for every super node in this CTCS, it first does the similar multiplication-addition operations as in CDG, and then transmits the M new obtained results to its parent in the subsequent level. Thus, to schedule a CTCS in the CDG way takes M super time slots instead of one. However, this way is more suitable for the pipeline operation by avoiding the data accumulation at nodes near the sink.

In summary, for CDC, ZPS pipeline the data transmission of $\lceil m/\omega \rceil$ continuous snapshots over $\lceil m/\omega \rceil$ segments with each segment transmits a snapshot respectively and concurrently. For a particular snapshot transmission within a segment, it is transmitted level by level by the CDG way. Finally, the sink can receive the data of a snapshot in every t_p super time slots after it receives the data of the first snapshot.

3.5.2 Capacity Analysis of ZPS

In this subsection, we analyze the the achievable data collection capacity of ZPS to collect N continuous snapshots. First, we investigate the consumed time slots to collect the first snapshot, which is the foundation of the formed data collection pipeline. Subsequently, we derive the achievable CDC capacity of ZPS in different cases.

Lemma 3.5.1 (i) *For the t_p in the inter-segment pipeline scheduling of ZPS, $t_p \leq \omega(2\omega - 1)M$; (ii) The number of super time slots used to collect the first snapshot is at most $\lceil \frac{m}{\omega} \rceil \omega(2\omega - 1)M$.*

Proof: (i) According to the intra-segment scheduling, the super nodes in each level of a segment can be partitioned into at most $2\omega - 1$ CTCSs. Moreover, for the super nodes within each CTCS, they transmit their data in the CDG way, i.e. each CTCS can be scheduled within M super time slots. Further, each segment contains at most ω levels, which implies for a single snapshot, a segment can be scheduled within $\omega(2\omega - 1)M$ super time slots, i.e. $t_p \leq \omega(2\omega - 1)M$.

(ii) Based on (i), the number of super time slots used to collect the first snapshot is at most $\lceil \frac{m}{\omega} \rceil \omega(2\omega - 1)M$, since a WSN can be partitioned into at most $\lceil \frac{m}{\omega} \rceil$ segments. □

Based on Lemma 3.5.1, we can derive the achievable CDC capacity of ZPS in different cases as shown in Theorem 3.5.1.

Theorem 3.5.1 *To collect N continuous snapshots, the achievable network capacity of ZPS is*

$$\begin{cases} \Omega(\frac{N\sqrt{n}}{6\sqrt{2}\omega M\sqrt{\log n}\ln n} \cdot W), & \text{if } N = O(\sqrt{n/\log n}); \\ \Omega(\frac{n}{12\omega^2 M \log n \ln n} \cdot W), & \text{if } N = \Omega(\sqrt{n/\log n}). \end{cases}$$

in the sense of the worst case, and

$$\begin{cases} \Omega(\frac{p_o N\sqrt{n/\log n}}{2\sqrt{2}\omega M} \cdot W), & \text{if } N = O(\sqrt{n/\log n}); \\ \Omega(\frac{p_o n}{4\omega^2 M \log n} \cdot W), & \text{if } N = \Omega(\sqrt{n/\log n}). \end{cases}$$

in the sense of expectation.

Proof: To collect N continuous snapshots, the data transmission process can be pipelined according to ZPS, which implies the sink can receive the data of a snapshot every t_p super time slots after it receives the first snapshot. Therefore, by Lemma 3.5.1, the number of super time slots used to collect N continuous snapshots is at most $\lceil \frac{m}{\omega} \rceil \omega(2\omega - 1)M + (N - 1)\omega(2\omega - 1)M \leq (\frac{m}{\omega} + 1) \cdot 2\omega^2 M + 2\omega^2(N - 1)M = O(2\omega mM + 2\omega^2 NM)$.

Thus, in the sense of the worst case, the achievable network capacity of ZPS is at least

$$\frac{NnB}{O(2\omega mM + 2\omega^2 NM) \cdot 10 \log n \cdot t_w} \tag{3.65}$$

$$= \frac{NnW}{O(20\omega mM\hbar \log n + 20\omega^2 \hbar NM \log n)} \tag{3.66}$$

$$= \frac{NnW}{O(10\omega M\hbar\sqrt{n \log n} + 20\omega^2 \hbar NM \log n)} \tag{3.67}$$

$$= \begin{cases} \Omega(\frac{N\sqrt{n}}{10\omega M\sqrt{\log n}\ln n} \cdot W), & \text{if } N = O(\sqrt{n/\log n}); \\ \Omega(\frac{n}{20\omega^2 M \log n \ln n} \cdot W), & \text{if } N = \Omega(\sqrt{n/\log n}). \end{cases} \tag{3.68}$$

Similarly, in the sense of expectation, the achievable network capacity of ZPS is at least

$$\frac{NnB}{O(2\omega mM + 2\omega^2 NM) \cdot 4\log n \cdot t_m} \tag{3.69}$$

$$= \frac{p_o NnW}{O(8\omega mM \log n + 8\omega^2 NM \log n)} \tag{3.70}$$

$$= \frac{p_o NnW}{O(4\omega M\sqrt{n \log n} + 8\omega^2 NM \log n)} \tag{3.71}$$

$$= \begin{cases} \Omega(\frac{p_o N\sqrt{n/\log n}}{4\omega M} \cdot W), & \text{if } N = O(\sqrt{n/\log n}); \\ \Omega(\frac{p_o n}{8\omega^2 M \log n} \cdot W), & \text{if } N = \Omega(\sqrt{n/\log n}). \end{cases} \tag{3.72}$$

□

From Theorem 3.5.1, we know that (*i*) the achievable network capacity of ZPS is $\frac{N\sqrt{n}}{\sqrt{\log n}\ln n}$ or $\frac{n}{\log n \ln n}$ times batter than the optimal capacity of the snapshot data collection scenario in order in the sense of the worst case, and $\sqrt{\frac{n}{\log n}}$ or $\frac{n}{\log n}$ times better than the optimal capacity of the snapshot data collection scenario in order in the sense of expectation, which are very significant improvements. By examining ZPS carefully, we find that two main reasons are responsible for this improvement. The primary reason is the use of the CDG technique, which distributes the traffic load evenly over the entire WSN, and then the data accumulation at the nodes near the sink is avoided. Another reason is the *pipeline scheduling*. According to ZPS, the time overlap of the data collection of multiple continuous snapshots in the transmission pipeline conserves a lot of time, which accelerates the network capacity directly and significantly; (*ii*) ZPS will be more effective for large-scale WSNs, since large scale WSNs incur large data collection trees, which are more suitable for pipeline; and (*iii*) ZPS is also more effective for long-term CDC. The longer the CDC process is, the closer for ZPS to its theoretical achievable network capacity.

3.6 Simulations

In this section, we validate the effectiveness of the proposed algorithms via simulations. For all the simulations, we consider a probabilistic WSN has one sink, and all the sensor

Table 3.2 System parameters.

Parameter	Value	Parameter	Value
α	3.0	$\frac{P}{N_0}$	10.0
η_1	0.25	ρ	3.0
η_2	10.0	M	50
η_3	10.0	N	1000

nodes are randomly distributed in a square area. The network time is slotted, and for the size of each time slot is normalized to one. Every node produces one data packet in a snapshot and the size of a packet is normalized to one. All the nodes work with the same power P over a common wireless channel, which has a bandwidth also normalized to one. Further, we define the node density of a WSN as ρ, i.e. ρ is the average number of nodes distributed within a unit area. In all the following simulations, we set $\rho = 3$. For the other parameters, we set them by referring the settings in [79] and they are given in Table 3.2. In Table 3.2, the parameters have the same meanings as in previous sections. As explained in Section *Network Model of Main File*, the success probability of a link between any two nodes can be obtained in terms of the parameters shown in Table 3.2 and *Equation 2 (Section Network Model) of the Main File*. Moreover, each group of simulations are repeated for 100 times and the results are the average of these 100 times.

Since there is no existing data collection algorithm for probabilistic WSNs currently, we compare our proposed algorithms with the latest data collection algorithms for deterministic wireless networks. The compared algorithms are PS [4] and MPS [5][59] for CPS.

- PS is the latest SDC algorithm based on a Breadth First Search (BFS) tree under the deterministic network model. First, PS constructs a BFS tree over the network graph. Subsequently, PS schedule the BFS tree path by path to collect the data on each path to the sink. By analysis, the authors showed that BFS can achieve order-optimal data collection capacity.

- MPS is also a SDC algorithm for deterministic WSNs, which extends PS to a multi-path data scheduling algorithm. In MPS, a CDS-based data collection tree is first

constructed. Then, multiple paths in the data collection tree are scheduled simultaneously as long as they are interference-free. By theoretical analysis, the authors demonstrated that MPS has a tighter capacity bound than that of PS.

- Although both PS and MPS are initially designed for deterministic WSNs, the algorithms themselves are actually independent of the underlying network model (deterministic or probabilistic). Therefore, when PS and MPS work in probabilistic WSNs, they follow the same original schedule idea. The only modification is that they may assign more than one time slots to a data transmission over a lossy link now, since a data transmission in probabilistic WSNs may have to transmit multiple times to guarantee that the receiver receives a data packet successfully. Furthermore, since PS and MPS are designed under the PrIM, we set the interference range of nodes in PS and MPS as $\omega \cdot l$ ($\omega \cdot l$ can prevent interference as we proven in Theorem 1).

For ZPS, we compare it with PSP (PS + pipeline) [4], CPSP (CPS + pipeline), CDGP (CDG + pipeline) [1], and PSA [5][59].

- PSP and CPSP are the pipelined versions of PS and PSP, respectively. CDGP is the pipelined version of CDG, which is a recent work for data collection for deterministic WSN and the first work applying the compressive sampling theory (the idea of CDG is discussed in Section *Continuous Data Collection of the Main File*). In PSP, CPSP, and CDGP, the data collection of each snapshot is scheduled in terms of PS, CPS, and CDG, respectively. Furthermore, the data collection of subsequent snapshots will be scheduled (also in terms of PS, CPS, and CDG) as soon as possible if their data transmissions are interference-free with the data transmissions of previous snapshots, i.e. in PSP, CPSP, and CDGP, subsequent snapshots may start to schedule for transmission before the sink receives previous snapshots.

  The reason to add the pipeline technique to PS and CDG is for fairness consideration.

- PSA is our previous work proposed for CDC in deterministic WSNs under the protocol interference model. In PSA, a CDS-based data collection tree is first constructed.

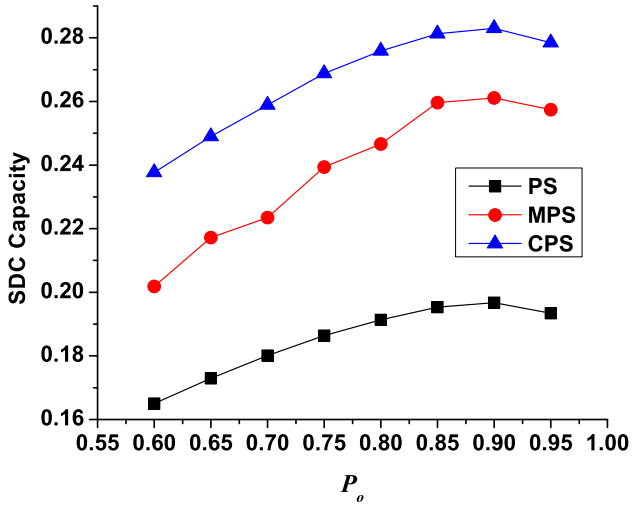


Figure 3.6 Snapshot data collection capacity.

Then, through partitioning the data collection tree into levels, a scheduling algorithm for CDC is designed exploiting pipeline and data compressing (CDG) techniques.

- Similarly, the designs of PSP, CDGP, and PSA are independent of the underlying network model (deterministic or probabilistic). Therefore, they can work in probabilistic WSNs by scheduling a data transmission over a lossy link until it is successfully finished.

3.6.1 Performance of CPS

We implement PS, MPS, and CPS in a probabilistic WSN deployed in an area of 100×100 for SDC, and the achievable capacities are shown in Figure 3.6 for different p_o values. From Figure 3.6 we know that when p_o varies from 0.6 to 0.9, the capacities of PS, MPS, and CPS increase. This is because a higher p_o implies fewer average transmissions over a lossy link (note that the average number of transmissions over a lossy link is $\frac{1}{p_o}$). Consequently, with the increasing of p_o, the capacities of PS, MPS, and CPS increase. However, when p_o varies from 0.9 to 0.95, the capacities of PS, MPS, and CPS decrease. This is because, on the other hand, a higher p_o also implies a larger R from the proof of Lemma 6. Whereas,

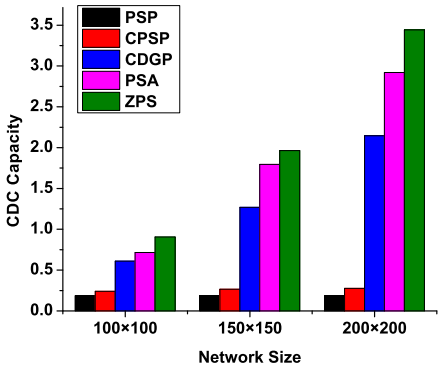
larger R implies fewer parallel transmissions can be concurrently conducted, which leads to the decrease of the achievable capacities of PS, MPS, and CPS. Note that although the capacities of CPS for the case $p_o = 0.8$ and the case $p_o = 0.95$ are similar, they have quite different meanings. Since small p_o implies more average transmission times, a network consumes less energy in the case $p_o = 0.95$ than that in the case $p_o = 0.8$ even they have similar capacities.

From Figure 3.6 we can also see that CPS always achieves a higher network capacity compared with PS and MPS. This is because that PS is a single path scheduling algorithm performed on a BFS tree. While CPS schedules a CTCS each super time slot, which is equivalent to schedule multiple cells on multiple paths. In other words, CPS achieves complete concurrency by scheduling multiple cells. Furthermore, the CDS-based data collection tree used by MPS is unbalanced and does not consider lossy links, which leads to the degradation of its capacity. Particularly, CPS achieves 44.02% more capacity than PS and 12% more capacity than MPS on average.

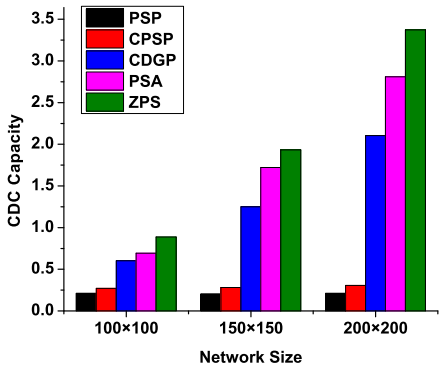
3.6.2 Performance of ZPS

To compare the performances of PSP, CPSP, CDGP, PSA, and ZPS for CDC, we conduct several groups of simulations in probabilistic WSNs with different sizes and p_o values, respectively. The results are shown in Figure 3.7. Specifically, in Figure 3.7(a)-(c), we fix the p_o value in each figure and compare the achievable network capacity of different algorithms in networks with different sizes. On the other hand, in Figure 3.7(d)-(f), we fix the network size in each figure and compare the achievable network capacity of different algorithms in networks with different p_o values.

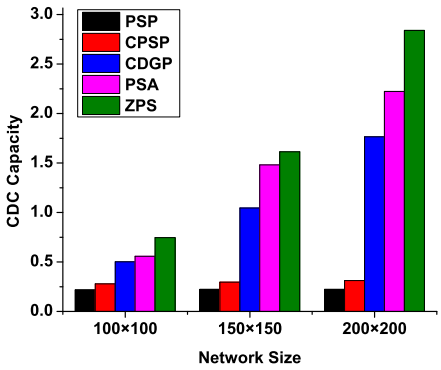
From Figure 3.7(a)-(c), we can see that with the increase of the network size (i.e. the number of nodes in a WSN), the achievable capacities of all the algorithms except for PSP and CPSP increase. This is because that the data transmission pipeline is easier to form and more effective in large-scale WSNs. On the other hand, since CDGP, PSA, and ZPS addressed the data accumulation problem, they can form effective data transmission



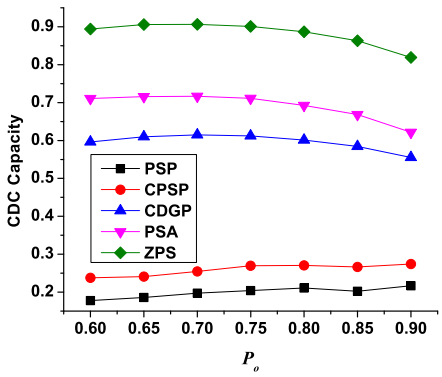
(a) $p_o = 0.65$.



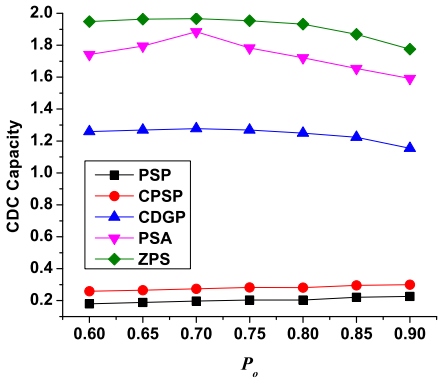
(b) $p_o = 0.8$.



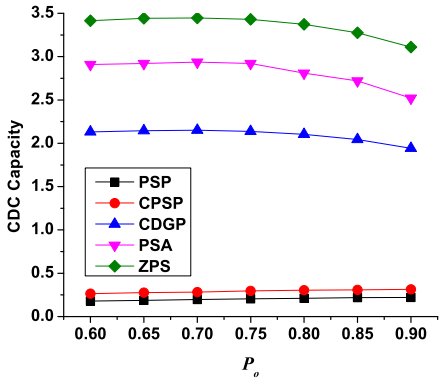
(c) $p_o = 0.95$.



(d) Network size: 100×100.



(e) Network size: 150×150.



(f) Network size: 200×200.

Figure 3.7 Continuous data collection capacity.

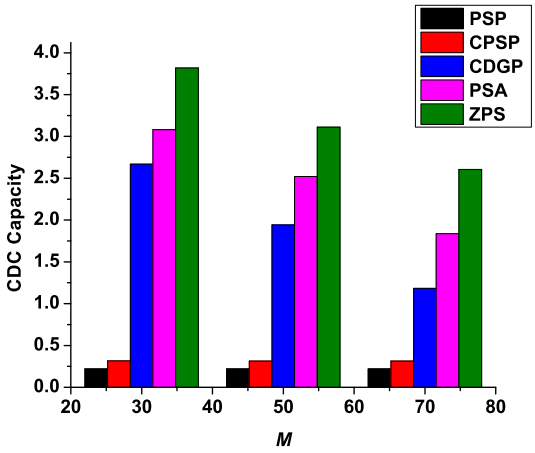
pipelines, which finally make them achieve higher capacities than PSP and CPSP. However, in PSP and CPSP, due to the existing of the data accumulation phenomenon near the sink, PSP and CPSP have similar data collection capacity in networks with different sizes.

From Figure 3.7(d)-(f), we can see that, because of the reasons discussed before, when p_o varying from 0.6 to 0.7, the capacities of CDGP, PSA, and ZPS have a slight increase. By contrast, when p_o varying from 0.75 to 0.9, the capacities of CDGP, PSA, and ZPS have some decrease. Additionally, the performance of CDGP, PSA, and ZPS highly depends on the data transmission pipeline. Larger p_o (i.e. larger R) implies larger segments (i.e. large t_p in Section *Continuous Data Collection of the Main File*), which further leads to the decrease of the capacities of CDGP, PSA, and ZPS. Moreover, since network size has a little impact on PS and CPS and data accumulates at nodes near the sink, PSP and CPSP also keep stable data collection capacities in WSNs with different sizes.

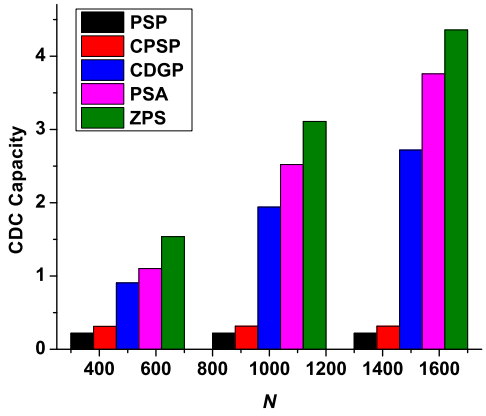
Finally, we can also see that ZPS achieves a higher capacity than CDGP and PSA from Figure 3.7(a)-(f). This is due to (*i*) when constructing data collection trees, PSA and CDGP do not consider lossy links; (*ii*) the data collection trees used by PSA and CDGP may be very unbalanced, which obstructs to form effective data transmission pipelines. By contrast, the data collection tree used by ZPS is balanced and has a more reasonable structure, which is more suitable to form a pipeline; (*iii*) ZPS has a more sound scheduling scheme compared with CDGP, i.e. a WSN is partitioned into multiple CTCSs, and ZPS achieves complete concurrency while scheduling these CTCSs.

3.6.3 Impacts of M and N on ZPS

The impacts of M (the parameter in CDG) and N (the number of snapshots in a CDC task) are shown in Figure 3.8. From Figure 3.8(a), we can see that the achievable capacities of CDGP, PSA, and ZPS decrease when M increases (PSA also exploits the CDG technique). This is because a large M implies more data packets have to be transmitted by each node for each snapshot. Consequently, more traffic are induced in CDGP, PSA, and ZPS, and followed by more time consumption and capacity degradation.



(a) Impacts of M ($N = 1000, p_o = 0.9$, and the network size is 200×200)



(b) Impacts of N ($M = 50, p_o = 0.9$, and the network size is 200×200)

Figure 3.8 Impacts of M and N on ZPS.

Figure 3.8(b) shows the impacts of N on the achievable capacity of ZPS. From Figure 3.8(b), the achievable capacities of CDGP, PSA, and ZPS increase with the increase of N. This is because that all the three algorithms finish CDC by forming data transmission pipeline systems, which prefer large network size and are more efficiency when the number of snapshots in a CDC task increases. This can also be seen from Theorem 3.

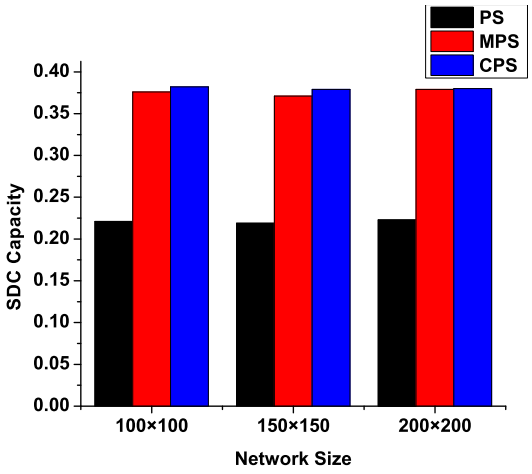
From Figure 3.8, we can also see that M and N almost have no impacts on the achievable capacities of PSP and CPSP. This is because that no data compressing technique is employed in the two algorithms. Moreover, PSP and CPSP do not provide dedicated solutions to the data accumulation problem in the CDC scenario, which is much severer than that in the SDC scenario.

3.6.4 CPS and ZPS in Deterministic WSNs

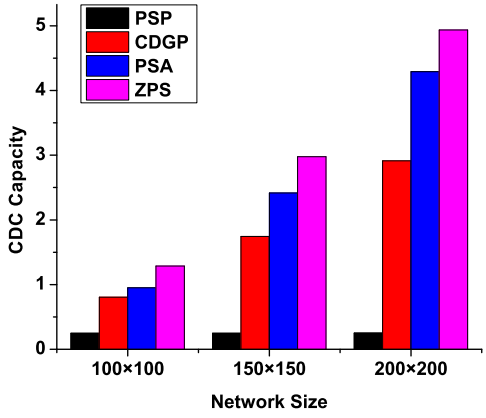
To examine the performance of CPS and ZPS in deterministic WSNs, we also implement them in deterministic WSNs with different sizes for completeness. The results are shown in Figure 3.9. From Figure 3.9, we know that even under the deterministic network model, CPS and ZPS achieve better data collection capacities compared with the existing works because of the subtle network partition and CTCS scheduling in CPS and ZPS.

3.7 Conclusion

For most existing works studying the network capacity issue, their designs and analysis are based on the deterministic network model. However, in real applications, this deterministic network model assumption is not practical due to the "transitional region phenomenon". Actually, a more practical network model for WSNs is the probabilistic network model, where a transmission over a link is conducted successfully with a probability instead of being determined. Unfortunately, few of the existing works study the data collection capacity issue for WSNs under the probabilistic network model, i.e. for probabilistic WSNs, until now. To fill in this gap, we investigate the achievable snapshot and CDC capacities for probabilistic WSNs in this part. For SDC, we propose a novel Cell-based Path Scheduling (CPS) algo-



(a) Performance of CPS in deterministic WSNs.



(b) Performance of ZPS in deterministic WSNs.

Figure 3.9 CPS and ZPS in deterministic WSNs.

rithm, which schedules multiple super nodes on multiple paths concurrently. Theoretical analysis of CPS shows that its achievable network capacity is order-optimal in the sense of expectation and has $O(\ln n)$ of degradation in the sense of the worst case. For CDC, we propose a *Zone-based Pipeline Scheduling* (ZPS) algorithm. ZPS significantly speeds up the CDC process by forming a data transmission pipeline, and achieves a surprising network capacity.The simulation results also validate that the proposed algorithms significantly improve network capacity compared with the existing works.

PART 4

DISTRIBUTED DATA COLLECTION IN ASYNCHRONOUS WIRELESS SENSOR NETWORKS

4.1 Introduction

Following the seminal work [19] by Gupta and Kumar, many works emerged to study the network capacity issue under various network scenarios, e.g. multicast, unicast, broadcast, and data collection/aggregation. However, to our knowledge, most of the existing works studied the network capacity issue under an ideal assumption that *the network time is slotted, and the entire network is strictly synchronized* explicitly or implicitly, i.e. they are mainly for *centralized synchronous* wireless networks. Under the above ideal assumption, many centralized algorithms with nice network capacity bounds are designed and analyzed for various communication modes (e.g. multicast, unicast, broadcast, and data collection/aggregation). In the sense of providing theoretical frameworks/bounds for the design of communication protocols, these works are still sound. However, in practice, wireless networks, especially WSNs, are more likely to be distributed systems. Furthermore, for WSNs, it is difficult and not realistic to achieve ideal strict time synchronization due to the unstable deployment environments, clock drift, and other technical limits. Therefore, to comprehensively and profoundly understand the performance of practical WSNs, it is important to investigate the achievable network capacity of *distributed asynchronous* WSNs. Particularly, we study the achievable data collection capacity for distributed asynchronous WSNs in this part.

Different from the study in centralized synchronous WSNs, there are many new challenges arising when investigating the data collection capacity issue for distributed asynchronous WSNs. We summarize the main challenges as follows.

- **C1:** unlike that in centralized synchronous WSNs, where we can acquire the overall information of a network and further make an optimized decision for data transmissions,

we can only schedule data transmissions according to local information in distributed asynchronous WSNs. Due to this reason, it is very difficult to find an optimal schedule. Therefore, how to design an effective distributed algorithm for data collection is a challenge.

- **C2:** since we cannot maintain a uniform time clock for all the sensor nodes in distributed asynchronous WSNs, every node carries out data transmissions based on its own time clock and local information. Intuitively, this kind of communication mode leads to many data collisions and retransmissions, incurring capacity degradation, unfairness among data flows, etc. Thus, how to avoid the disadvantages introduced by an asynchronous time scheme is a primary concern when designing distributed data collection algorithms.

- **C3:** following challenges **C1** and **C2**, the third challenge is how to theoretically analyze the achievable network capacity bounds for a data collection algorithm in distributed asynchronous WSNs. Since the data collection algorithm works in a distributed manner, it is difficult, sometimes even impossible, to know the exact time when a data transmission occurs, as well as the time duration of a data transmission. Hence, both elegant analysis techniques and a carefully designed data transmission mechanism are important to obtain the achievable data collection capacity.

To address these challenges, we propose a scalable and order-optimal distributed algorithm, named *Distributed Data Collection* (DDC), with fairness consideration and capacity analysis under the *generalized physical interference model*. To the best of our knowledge, this is the first attempt to provide detailed protocol design and rigorous capacity analysis for data collection in distributed asynchronous WSNs. DDC works in a CSMA-like manner, except for the RTS/CTS communication mode and the necessity to reply an ACK packet after receiving a data packet. In DDC, when a sensor node has some data packets for transmission, it sets up a backoff timer, and senses the wireless channel with a predefined *Carrier-sensing Range* (CR). If the channel is free when the backoff timer expires, this node

conducts a data transmission. Under this transmission manner, DDC gathers all of the data in a network to the sink (i.e. base station). Moreover, we extend our data collection method to the case of data gathering with aggregation, and propose a *Distributed Data Aggregation* (DDA) algorithm. We summarize the main contributions of this part as follows.

1. The carrier-sensing range is an important parameter in DDS, which has a significant impact on the performance of data collection. To avoid data transmission collisions/interference, especially the collisions/interference caused by the *hidden-node problems*, we derive an $\mathcal{R}_0$-*Proper Carrier-sensing Range* ($\mathcal{R}_0$-PCR) under the *generalized physical interference model* for the nodes in a data collection WSN, where $\mathcal{R}_0$ is the *satisfied threshold of data receiving rate*. By taking $\mathcal{R}_0$-PCR as its CR, any node can initiate a data transmission with guaranteed data receiving rate as long as there is no ongoing transmissions within its CR.

2. Based on the obtained $\mathcal{R}_0$-PCR, we propose a scalable and order-optimal *Distributed Data Collection* (DDC) algorithm with fairness consideration for asynchronous WSNs. DDC works in a CSMA-like manner, and effectively gathers all the data to the sink. Theoretical analysis of DDC surprisingly shows that its asymptotic achievable network capacity is $\mathbb{C} = \Omega(\frac{1}{2(\beta_\kappa+\beta_{\kappa+1})} \cdot W)$, where β_x ($x \in \{\kappa, \kappa+1\}$) is a constant value depends on $\mathcal{R}_0$, and W is the bandwidth of a wireless communication channel. Since the upper bound capacity of data collection is $O(W)$ [4][5], which implies the achievable data collection capacity of DDC is order-optimal. Furthermore, since $\mathbb{C}$ is independent of network size, DDC is scalable.

3. For completeness, a *Distributed Data Aggregation* (DDA) algorithm for asynchronous WSNs is designed. We show that the number of time slots induced by DDA is upper bounded by $\log n + (\beta_\kappa + \beta_{\kappa+1} - 1)L + c_3$, where n is the number of the sensor nodes in a WSN, L is the hight of the data aggregation tree, and c_3 is a constant value depending on $\mathcal{R}_0$-PCR.

4. To be more general, we further study the delay and capacity of DDC and DDA under

the Poisson node distribution model. By analysis, we demonstrate that DDC is again scalable and order-optimal, and DDA has a delay performance upper bounded by $a \log n + (\beta_\kappa + \beta_{\kappa+1} - 1)L - c_4$, where $a = \arg\min_{\epsilon>0} (\frac{2}{\epsilon} + \frac{\pi\lambda R^2(e^\epsilon - 1)}{\epsilon \log n})$ and $c_4 = \beta_\kappa + \beta_{\kappa+1}$ are constant values.

5. We also conduct extensive simulations to validate the performance of DDC/DDA in distributed asynchronous WSNs. The simulation results indicate that DDC/DDA can achieve comparable data collection capacity as the latest centralized and synchronized data collection algorithm.

The rest of this part is organized as follows. In Section 4.2, the considered network model is discussed. In Section 4.3, the proper carrier-sensing range satisfying a predefined data receiving rate for communication is derived. According to the obtained proper carrier-sensing range, a distributed asynchronous data collection algorithm is proposed in Section 4.4, followed by the theoretical analysis, which demonstrates that the proposed algorithm can achieve order-optimal data collection capacity as centralized and synchronized algorithms. Furthermore, how to applying the derived proper carrier-sensing range to data aggregation is discussed in Section 4.5. To be more general, we study the delay and capacity of DDC and DDA under the Poisson distribution model in Section 4.6. In Section 4.7, we validate the performance and scalability of DDC and DDA by simulations. Finally, this part is concluded and some possible future research directions are pointed out in Section 4.8.

4.2 Network Model

In this part, we consider a connected WSN consisting of one sink node serving as the *base station* denoted by s_0, and n sensor nodes denoted by $s_1, s_2, \cdots, s_n$ respectively, deployed in an area with size $A = c_1 n$, where c_1 is a constant. Furthermore, we assume all the nodes are *independent and identically distributed* (*i.i.d.*). Each node is equipped with one radio and works with a fixed power P. All the data transmissions are conducted over a common wireless channel with bandwidth W *bits/second*. The size of a data packet is B bits, and thus

the transmission duration of a data packet is $\tau = B/W$ seconds. The maximum transmission radius of a sensor node is set to r (r is associated with the lowest data transmission rate determined by the following defined *generalized physical interference model*). Hence the network can be modeled as a graph $G = (V, E)$, where $V = \{s_i | i = 0, 1, 2, \cdots, n\}$ and E includes all the possible links formed by any pair of nodes in V. A node s_i ($i \in [1, n]$) is said to be *active* at time t *iff* s_i is transmitting a data packet to some other node at time t. Thus, we use $\mathcal{S}^t = \{s_k | s_k \textit{ is active at time } t\}$ to denote the set of all the active nodes at time t.

To capture the wireless interference in wireless networks, the *protocol interference model* and *physical interference model* are frequently used. Furthermore, these two models abstract a data transmission as a binary function, with values *successful* or *failed*. Instead of modeling a data transmission process as a binary function, the *Generalized Physical Interference model* (GPI) is more accurate to characterize a practical data transmission. Suppose node s_i is transmitting a data packet to node s_j at time t, i.e. $s_i \in \mathcal{S}^t$, and $\mathcal{R}^t_{i,j}$ is the data receiving rate of s_j from s_i at time t. Then, under the GPI model, $\mathcal{R}^t_{i,j}$ is determined by

$$\mathcal{R}^t_{i,j} = W \cdot \log(1 + SINR^t_{i,j}) \tag{4.1}$$

where $SINR^t_{i,j}$ is the *Signal-to-Interference-plus-Noise Ratio* (SINR) value at s_j associated with s_i and is defined as

$$SINR^t_{i,j} = \frac{P \cdot D(s_i, s_j)^{-\alpha}}{N_0 + \sum\limits_{s_k \in \mathcal{S}^t, s_k \neq s_i} P \cdot D(s_k, s_j)^{-\alpha}} \tag{4.2}$$

where N_0 is the background noise, α is the path loss exponent and usually $\alpha \geq 3$, and $D(\cdot, \cdot)$ is the Euclidian distance between two nodes.

Suppose the time consumption to gather all the n data packets produced at s_i ($1 \leq i \leq n$) is $\mathcal{T}$, then the achievable data collection capacity $\mathbb{C}$ can be defined as $nB/\mathcal{T}$, i.e. the data collection capacity reflects how fast that data can be gathered by the sink.

4.3 Carrier-sensing Range

Since we study data collection in distributed asynchronous WSNs, every node s_i ($i \in [1, n]$) in a WSN senses the activities of other nodes within its *Carrier-sensing Range* (CR) when it has some data packets for transmission. Only when there is no ongoing data transmissions within its CR, s_i can initiate a data transmission. Thus, how to determine the CR for each node, to make all the concurrent transmitters out of the CR of each other to simultaneously conduct data transmissions with a data rate no less than a threshold, is crucial for the performance of a distributed data collection scheme. Intuitively, a small CR implies a high degree of spatial reuse, which further implies small SINR values and followed by low data receiving rates at the receivers. On the other hand, a large CR implies a low degree of spatial reuse, which further implies large SINR values and high data receiving rates. Therefore, in this section, we study how to set a *Proper Carrier-sensing Range* (PCR) for each node to guarantee a satisfied data receiving rate and meanwhile the highest spatial reuse degree. For clarity, we make some definitions as follows.

Definition 4.3.1 ***$\mathcal{R}_0$-feasible state.*** *The set of all the active nodes $\mathcal{S}^t$ (defined in Section 4.2) is an $\mathcal{R}_0$-feasible state if all the nodes in $\mathcal{S}^t$ can simultaneously transmit data and the data receiving rate at each of their corresponding receivers is no less than $\mathcal{R}_0$. In an $\mathcal{R}_0$-feasible state $\mathcal{S}^t$, $\forall s_i \in \mathcal{S}^t$, assume s_i is transmitting a data packet to s_j, then $\mathcal{R}^t_{i,j} \geq \mathcal{R}_0$.*

Based on Definition 4.3.1, if the lowest tolerable data transmission rate of a WSN is $\mathcal{R}_0$, then the data collection process can be represented as a series of $\mathcal{R}_0$-feasible states $\mathcal{S}^t$ ($t = \tau, 2\tau, 3\tau, \cdots, m\tau$), where $m = \lceil \mathcal{T}/\tau \rceil$.

Definition 4.3.2 ***R-set ($\mathcal{S}_R$).*** *Assume R is the carrier-sensing range represented by $G = (V, E)$. An R-set, denoted by $\mathcal{S}_R$, is any maximal subset of V that satisfies $\forall s_i, s_j \in \mathcal{S}_R$ ($s_i \neq s_j$) and $D(s_i, s_j) \geq R$.*

Definition 4.3.3 ***$\mathcal{R}_0$-Proper Carrier-sensing Range ($\mathcal{R}_0$-PCR).*** *The carrier-sensing range R of a WSN is an $\mathcal{R}_0$-proper carrier-sensing range if for any R-set $\mathcal{S}_R$, it is always an $\mathcal{R}_0$-feasible state.*

From Definition 4.3.3, if R is an $\mathcal{R}_0$-PCR, then s_i can initiate a data transmission with a guaranteed data receiving rate no less than $\mathcal{R}_0$ as long as there is no other active nodes within R of s_i. Then, given a threshold of data receiving rate $\mathcal{R}_0$, the $\mathcal{R}_0$-PCR can be determined by the following Theorem 4.3.1. In the following analysis, as that in [80], we assume the background noise is very small compared with the transmission power ($N_0 \ll P$) and thus can be ignored.

Theorem 4.3.1 *$\mathcal{R}_0$-PCR $\geq (\sqrt[\alpha]{c_2(2^{\mathcal{R}_0/W}-1)}+1)\cdot r$, where c_2 is a constant.*

Proof: Let $R = \mathcal{R}_0$-PCR and $I = R - r$. To make any R-set $\mathcal{S}_R$ always an $\mathcal{R}_0$-feasible state, for $\forall s_i \in \mathcal{S}_R$, assuming its destination node is s_j, then, we have

$$\mathcal{R}_{i,j} \geq \mathcal{R}_0 \tag{4.3}$$

$$\Leftrightarrow W \cdot \log(1 + SINR_{i,j}) \geq \mathcal{R}_0 \tag{4.4}$$

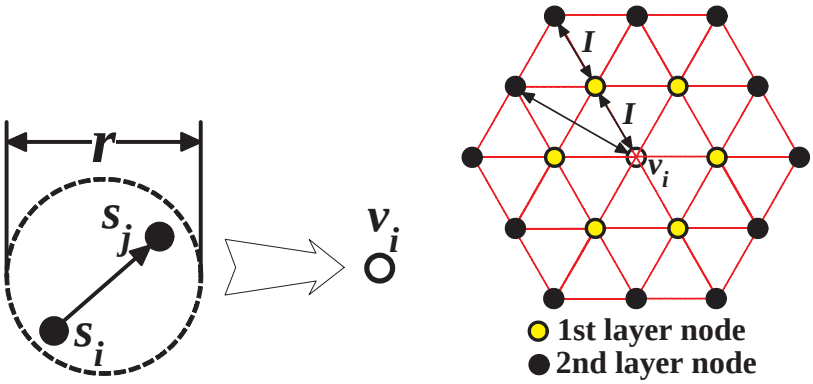
$$\Leftrightarrow 1 + SINR_{i,j} \geq 2^{\mathcal{R}_0/W} \tag{4.5}$$

$$\Leftrightarrow SINR_{i,j} \geq 2^{\mathcal{R}_0/W} - 1 \tag{4.6}$$

$$\Leftrightarrow \frac{P \cdot D(s_i, s_j)^{-\alpha}}{N_0 + P \cdot \sum\limits_{s_k \in \mathcal{S}_R, s_k \neq s_i} D(s_k, s_j)^{-\alpha}} \geq 2^{\mathcal{R}_0/W} - 1 \tag{4.7}$$

$$\Leftrightarrow \frac{D(s_i, s_j)^{-\alpha}}{\sum\limits_{s_k \in \mathcal{S}_R, s_k \neq s_i} D(s_k, s_j)^{-\alpha}} \geq 2^{\mathcal{R}_0/W} - 1 \tag{4.8}$$

Now, we derive the lower bound of $\frac{D(s_i,s_j)^{-\alpha}}{\sum\limits_{s_k \in \mathcal{S}_R, s_k \neq s_i} D(s_k,s_j)^{-\alpha}}$. Evidently, $D(s_i, s_j)^{-\alpha} \geq r^{-\alpha}$ since r is the maximum transmission range of a node (defined in Section 4.2). Furthermore, if we abstract a data transmission link as a node as shown in Figure 4.1(a), then, for the nodes in $\mathcal{S}_R$, the densest packing of nodes is the hexagon packing [80] with edge length I as shown in Figure 4.1(b). Subsequently, the nodes in $\mathcal{S}_R$ can be layered with respect to v_i



(a) (b)

Figure 4.1 (a) Link abstraction and (b) hexagon packing.

(abstracted by the transmission link from s_i to s_j), with the l-th layer having at most $6l$ nodes. Furthermore, the distance between v_i and any node at the l-th layer is no less than $\frac{\sqrt{3}}{2}lI$. Then, we have

$$\sum_{s_k \in \mathcal{S}_R, s_k \neq s_i} D(s_k, s_j)^{-\alpha} \tag{4.9}$$

$$\leq 6 \cdot I^{-\alpha} + \sum_{l \geq 2} 6l \cdot (\frac{\sqrt{3}}{2} lI)^{-\alpha} \tag{4.10}$$

$$= 6 \cdot I^{-\alpha} + 6 \cdot (\frac{\sqrt{3}}{2} I)^{-\alpha} \cdot \sum_{l \geq 2} l^{-\alpha+1} \tag{4.11}$$

In Equation 4.11, $\sum_{l \geq 2} l^{-\alpha+1} = \zeta(\alpha - 1) - 1$, where $\zeta(\cdot)$ is the *Riemann zeta function*. Considering that $\alpha \geq 3$, then $\zeta(\alpha - 1) \leq \zeta(2) = \frac{\pi^2}{6}$. It follows that $\sum_{l \geq 2} l^{-\alpha+1} \leq \frac{\pi^2}{6} - 1$. Thus, we have

$$\sum_{s_k \in \mathcal{S}_R, s_k \neq s_i} D(s_k, s_j)^{-\alpha} \tag{4.12}$$

$$\leq 6 \cdot I^{-\alpha} + 6 \cdot (\frac{\sqrt{3}}{2} I)^{-\alpha} \cdot (\frac{\pi^2}{6} - 1) \tag{4.13}$$

$$= (6 + (\pi^2 - 6)(\frac{\sqrt{3}}{2})^{-\alpha}) \cdot I^{-\alpha} \tag{4.14}$$

$$= c_2 \cdot I^{-\alpha}, \tag{4.15}$$

where $c_2 = (6 + (\pi^2 - 6)(\frac{\sqrt{3}}{2})^{-\alpha})$. It follows that

$$\frac{D(s_i, s_j)^{-\alpha}}{\sum_{s_k \in \mathcal{S}_R, s_k \neq s_i} D(s_k, s_j)^{-\alpha}} \geq \frac{r^{-\alpha}}{c_2 \cdot I^{-\alpha}}. \tag{4.16}$$

Therefore, to make Equation 4.8 valid, it is sufficient to have

$$\frac{r^{-\alpha}}{c_2 \cdot I^{-\alpha}} \geq 2^{\mathcal{R}_0/W} - 1 \tag{4.17}$$

$$\Leftrightarrow I^{-\alpha} \leq \frac{r^{-\alpha}}{c_2(2^{\mathcal{R}_0/W} - 1)} \tag{4.18}$$

$$\Leftrightarrow I \geq (\frac{1}{c_2(2^{\mathcal{R}_0/W} - 1)})^{-1/\alpha} \cdot r \tag{4.19}$$

$$\Leftrightarrow I \geq \sqrt[\alpha]{c_2(2^{\mathcal{R}_0/W} - 1)} \cdot r. \tag{4.20}$$

Therefore, $\mathcal{R}_0$-PCR $= R = I + r \geq \sqrt[\alpha]{c_2(2^{\mathcal{R}_0/W} - 1)} \cdot r + r = (\sqrt[\alpha]{c_2(2^{\mathcal{R}_0/W} - 1)} + 1) \cdot r$. □

From Theorem 4.3.1, we know that given a threshold of data receiving rate $\mathcal{R}_0$, we can determine an $\mathcal{R}_0$-PCR, which is at least a constant times r. Since a small CR implies a high degree of spatial reuse, we set $\mathcal{R}_0$-PCR $= (\sqrt[\alpha]{c_2(2^{\mathcal{R}_0/W} - 1)} + 1) \cdot r$. Furthermore, Figure 4.2 depicts the relation between $\mathcal{R}_0$ and $\mathcal{R}_0$-PCR, where the X-axis represents the threshold of data receiving rate $\mathcal{R}_0$, and the Y-axis represents the corresponding $\mathcal{R}_0$-PCR. From Figure 4.2, we can tell with the increase of $\mathcal{R}_0$, the associated $\mathcal{R}_0$-PCR increases accordingly for every α value. This is because a high data receiving rate requires that CR should be sufficiently large to avoid interferences, which also implies a low degree of spatial reuse. Additionally, a large α also implies a small $\mathcal{R}_0$-PCR. This is because the interference impact decreases quickly with the increase of α, which can also be derived from Equation 4.2.

4.4 Distributed Data Collection and Capacity

According to the obtained $\mathcal{R}_0$-PCR in Section 4.3, if we set the CR of a WSN as $\mathcal{R}_0$-PCR, then all the nodes in an R-set ($R = \mathcal{R}_0$-PCR) can simultaneously transmit data

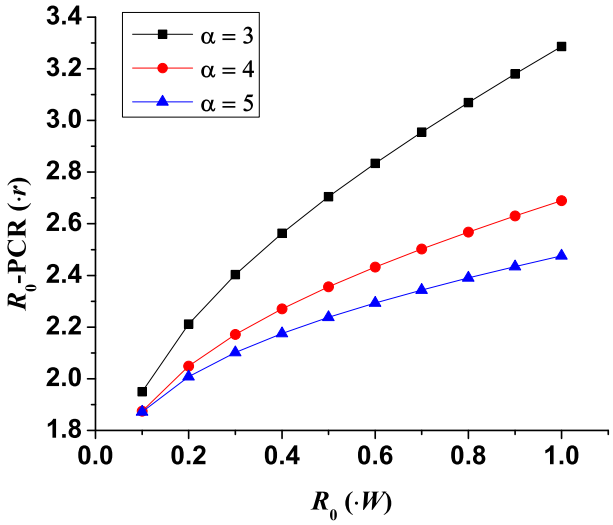


Figure 4.2 $\mathcal{R}_0$ *vs.* $\mathcal{R}_0$-PCR.

at a guaranteed data receiving rate without interference by letting each node work on the Re-Start (RS) mode [80]. Thus, in this section, we propose a CSMA-like data collection algorithm for distributed asynchronous WSNs, which has an order-optimal capacity.

4.4.1 Distributed Data Collection

Before presenting the distributed data collection algorithm, for a WSN represented by $G = (V, E)$, we construct a *Connected Dominating Set* (CDS)-based data collection tree, denoted by T, according to the method in [7]. The construction process is discussed in Part 2.

Assume L is the height of T, i.e. the maximum number of hops from s_0 to any node, and $L(s_i)$ is the number of hops from node s_i to s_0 in T. Evidently, according to the construction process of T, $\forall s_i \in \mathcal{D}$, $L(s_i)$ is an even number, and $\forall s_j \in \mathcal{C}$, $L(s_j)$ is an odd number. Furthermore, we define $\mathcal{L}_\iota = \{s_i | L(s_i) = \iota\}$ $(0 \leq \iota \leq L)$. Then, the following lemma [7] shows some properties of T.

Lemma 4.4.1 [7] *(i)* s_0 *is adjacent to at most* 12 *connectors in* $\mathcal{C}$*; (ii)* $\forall s_i \in \mathcal{D}, s_i \neq s_0$, s_i *is adjacent to at most* 11 *connectors in* $\mathcal{L}_{L(s_i)+1}$.

Based on T, we propose a *Distributed Data Collection* (DDC) algorithm for asynchronous WSNs as shown in Algorithm **??**. In Algorithm **??**, $counter(s_i)$ is a counter that denotes the number of data packets transmitted by s_i, τ_w is the *backoff contention window*, and t_i^j $(1 \leq j \leq counter(s_i))$ is the backoff time set for the transmission of the j-th data packet at node s_i. As that in [80] and because of the same reasons, we assume (i) $\tau_w \ll \tau$ such that τ_w is negligible compared with the data transmission time, and (ii) no two transmitters within the CR of each other have their backoff timers expired at the same time instant[1].

According to Algorithm **??**, DDC runs in a CSMA-like manner, except for the RTS/CTS working mode and the necessity to reply an ACK packet after receiving a data packet. This is because that by properly setting the CR and working in the RS mode, a transmission with satisfied data receiving rate can be guaranteed as shown in Section 4.3.

In Algorithm **??** (here, taking the algorithm running process at node s_i as an example), Lines 1-5 are basic settings. Line 6 randomly sets the backoff time for each data transmission. In Lines 7-8, the backoff time for each transmission is reset to $(\tau_w - t_i^{j-1}) + t_i^j$, and this is mainly for fairness (any node will not wait too long when it has some data to transmit) as shown in Theorem 4.4.1 and Corollary 4.4.2 (see Section 4.4.2). Under this setting, a node cannot transmit multiple data packets in a short time period. Actually, each node can transmit up to one data packet during each backoff contention window. In Lines 9-14, s_i begins the countdown process and keeps sensing the channel with $\mathcal{R}_0$-PCR. If the wireless channel is busy sensed by s_i, the countdown process at s_i will be frozen. In this way, when a data transmission is ongoing, all the other nodes having data packets within the CR of the transmitter will stop their countdown process, i.e. they can share the waiting time. In Lines 15-16, s_i transmits the j-th data packet when the backoff timer expires. Since no two transmitters that within the CR of each other have their backoff timers expired at the same time instant, the transmission of the j-th data packet can carried out successfully.

[1]Collisions due to simultaneous countdown-to-zero can be tackled by an exponential backoff mechanism in which the transmission probability of each node is adjusted in a dynamic way based on the network busyness [80].

Algorithm 3: The DDC Algorithm

input : CDS-based data collection tree T, $\mathcal{R}_0$-PCR
output: a distributed asynchronous data collection plan

```
1  counter(s_i)←0;
2  s_i(i ∈ [1, n]) sets its CR as R_0-PCR according to the required threshold of data
   receiving rate R_0;
3  while s_i has some data packets for transmission do
4      counter(s_i)← counter(s_i) + 1;
5      j ←counter(s_i);
6      s_i randomly sets a backoff time t_i^j for the transmission of the j-th packet in
       window (0, τ_w];
7      if j > 1 then
8          t_i^j ← (τ_w − t_i^{j−1}) + t_i^j;
9      while t_i^j is not countdown to 0 do
10         s_i senses the channel with R_0-PCR;
11         if s_i senses that the channel is busy then
12             s_i stops the countdown process (the backoff timer is frozen) until the
               channel becomes free again;
13         if s_i senses that the channel is free then
14             t_i^j − −;
15     if t_i^j == 0, i.e. the backoff timer expires then
16         s_i transmits the j-th data packet to its parent node;
```

4.4.2 Capacity Analysis

In this subsection, we analyze the achievable data collection capacity of the DDC algorithm. Since the upper bound capacity of data collection is $O(W)$ [4][5], we investigate the lower bound capacity of DDC in this subsection. First, we study the upper bound time consumption to collect all data packets at dominatees to the CDS, i.e. the upper bound time consumption to collect data packets at $V \setminus (\mathcal{D} \cup \mathcal{C})$ to $\mathcal{D} \cup \mathcal{C}$.

Let $R = \mathcal{R}_0$-PCR $= (\sqrt[\alpha]{c_2(2^{\mathcal{R}_0/W} - 1)} + 1) \cdot r$, where $\mathcal{R}_0$-PCR is the CR used in DDC. Then, we have the following lemma which indicates the average/upper bound number of the sensor nodes, denoted by $\mathbb{A}/\mathbb{U}$, within the CR of a node.

Lemma 4.4.2 *Let the random variable X denote the number of sensor nodes within the carrier-sensing area of a node. Then,*

(i) $\mathbb{A} = \mathbf{E}[X] = \frac{\pi R^2}{c_1}$.

(ii) $\Pr[X > \log n + \frac{\pi R^2(e^2-1)}{2c_1}] \leq \Pr[X \geq \log n + \frac{\pi R^2(e^2-1)}{2c_1}] \leq \frac{1}{n^2}$. *Thus, it is almost impossible that the carrier-sensing area of a node contains more than* $\log n + \frac{\pi R^2(e^2-1)}{2c_1}$ *nodes, i.e. it is almost sure that* $\mathbb{U} = \log n + \frac{\pi R^2(e^2-1)}{2c_1}$.

Proof: Since all the wireless nodes are *i.i.d.* in an area with size $A = c_1 n$, then for any node, it is located at the carrier-sensing area of a particular node with probability $p = \frac{\pi R^2}{c_1 n}$. Then, X satisfies the *binomial distribution* with parameters (n, p). Thus, the average number of the nodes within the carrier-sensing area of a node is $\mathbb{A} = np = \frac{\pi R^2}{c_1}$.

Now, we prove the second statement. Let $a = \log n + \frac{\pi R^2(e^2-1)}{2c_1}$. Then, applying the

Chernoff bound and for any $\epsilon > 0$, we have

$$\Pr[X > a] \leq \Pr[X \geq a] \tag{4.21}$$
$$\leq \min_{\epsilon>0} \frac{\mathbf{E}[e^{\epsilon X}]}{e^{\epsilon a}} \tag{4.22}$$
$$= \min_{\epsilon>0} \frac{[1 + (e^{\epsilon} - 1)p]^n}{e^{\epsilon a}} \tag{4.23}$$
$$\leq \min_{\epsilon>0} \frac{e^{(e^{\epsilon}-1)pn}}{e^{\epsilon a}} \tag{4.24}$$
$$= \min_{\epsilon>0} \exp[(e^{\epsilon} - 1)pn - \epsilon a] \tag{4.25}$$
$$= \min_{\epsilon>0} \exp[(e^{\epsilon} - 1)\mathbb{A} - \epsilon a]. \tag{4.26}$$

Particularly, let $\epsilon = 2$, then

$$\Pr[X > a] \tag{4.27}$$
$$\leq \exp[(e^2 - 1)\mathbb{A} - 2a] \tag{4.28}$$
$$= \exp[(e^2 - 1) \cdot \frac{\pi R^2}{c_1} - 2(\log n + \frac{\pi R^2(e^2 - 1)}{2c_1})] \tag{4.29}$$
$$= \exp[-2 \log n] \leq \exp[-2 \ln n] \tag{4.30}$$
$$= \frac{1}{n^2}. \tag{4.31}$$

$\sum_{n>0} \frac{1}{n^2}$ is the *Riemann zeta function* with parameter 2, and $\sum_{n>0} \frac{1}{n^2} = \frac{\pi^2}{6} < \infty$. It follows that $\Pr[X \leq a] \approx 1$ according to the Borel-Cantelli Lemma, i.e. it is almost sure that the carrier-sensing area of a node contains no more than $\log n + \frac{\pi R^2(e^2-1)}{2c_1}$ nodes. Thus, it is reasonable to use $\log n + \frac{\pi R^2(e^2-1)}{2c_1}$ as the upper bound of the number of the nodes within the carrier-sensing area of a node, i.e. $\mathbb{U} = \log n + \frac{\pi R^2(e^2-1)}{2c_1}$. □

Based on Lemma 4.4.2, we can derive the upper bound time consumption to collect all the data packets at $V \setminus (\mathcal{D} \cup \mathcal{C})$ to $\mathcal{D} \cup \mathcal{C}$ in DDC.

Theorem 4.4.1 *Any node s_i with data packets for transmission can transmit at least one data packet to its parent node within time $2\mathbb{U}\tau = (2 \log n + \frac{\pi R^2(e^2-1)}{c_1})\tau$.*

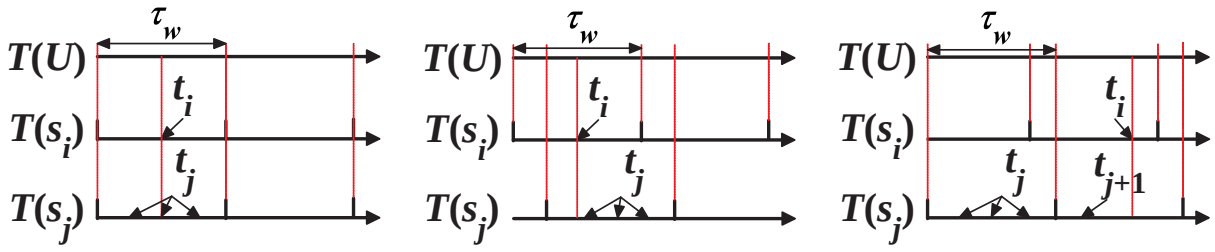


Figure 4.3 Transmission sequence of s_i and s_j.

Proof: According to the DDC algorithm, for any node s_i with data packets for transmission, it will carrier-senses the node activities within its CR. When the backoff timer of s_i expires and meanwhile the channel sensed by s_i is free, s_i can transmit a data packet successfully. Thus, the problem now is how long it takes for s_i until it actually initiates a data transmission in the worst case, i.e. the waiting time of s_i in the worst case. For convenience, assume s_j is any other node within the CR of s_i having data packets for transmission, $t_i, t_j \in (0, \tau_w](t_i \neq t_j)$ are the backoff time for the current data transmissions of s_i and s_j respectively, and $T(U)$, $T(s_i)$ and $T(s_j)$ are the universal time (standard time), the system time maintained at s_i and s_j respectively. Furthermore, if s_j has more than one data packet for transmission, the backoff time for s_j to transmit a subsequent data packet is denoted by t_{j+1}. Evidently, the transmission sequence of s_i and s_j follows one of the following three cases (Note that no two transmitters within the CR of each other have their backoff timers expired at the same time instant).

Case 1: s_i and s_j share a synchronized backoff contention window. In this case, as shown in Figure 4.3(a), s_i will transmit a data packet before/after s_j transmits a data packet. This is because $t_{j+1} = t_j + (\tau_w - t_j) + t'_{j+1} = \tau_w + t'_{j+1} > t_i$, where $t'_{j+1} \in (0, \tau]$ is the backoff time chosen by s_j for the subsequent data transmission according to the DDC algorithm.

Case 2: s_i and s_j share an asynchronous backoff contention window and $t_i < t_j$. In this case, as shown in Figure 4.3(b), s_i will transmit a data packet before s_j according to DDC.

Case 3: s_i and s_j share an asynchronous backoff contention window and $t_i > t_j$. In this

case, as shown in Figure 4.3(c), when s_i tries to transmit a data packet, it sets a backoff time t_i for the packet and carrier-senses the channel. It turns out that the channel is busy since s_j is transmitting some data. Therefore, we conclude that $0 < t_i - t_j < 2\tau_w$ because the time slots of s_i and s_j have some overlap (otherwise, s_i cannot know that the channel is occupied by s_j when it tries to transmit the data packet). Since $0 < t_i - t_j < 2\tau_w$, it is possible that $t_{j+1} = t_j + (\tau_w - t_j) + t'_{j+1} = \tau_w + t'_{j+1} < t_i$. This implies that s_j may transmit two data packets before s_i transmits one data packet. On the other hand, according to the DDC algorithm, we have $t_{j+2} = t_j + (\tau_w - t_j) + t'_{j+1} + (\tau_w - t'_{j+1}) + t'_{j+2} = 2\tau + t'_{j+2} > t_i$, where t_{j+2} is the time that s_j transmits its third data packet and t'_{j+2} is the backoff time set by s_j for its third data packet transmission. Consequently, s_i will transmit one data packet before s_j transmits the third data packet.

In summary, s_j can transmit at most two data packets before s_i transmits one data packet in the worst case. Considering that there are at most $\mathbb{U}$ sensor nodes within the carrier-sensing area of s_i according to Lemma 4.4.2, s_i can transmit at least one data packet to its parent node within time $2\mathbb{U}\tau$ in the worst case in DDC. □

Corollary 4.4.1 *In DDC, the time consumption of collecting all the data packets at* $V \setminus (\mathcal{D} \cup \mathcal{C})$ *to* $\mathcal{D} \cup \mathcal{C}$ *is at most* $2\mathbb{U}\tau$.

Proof: Based on the construction process of the data collection tree T, every node in $V \setminus (\mathcal{D} \cup \mathcal{C})$ has a parent node in $\mathcal{D} \cup \mathcal{C}$. Thus, all the data packets at $V \setminus (\mathcal{D} \cup \mathcal{C})$ can be transmitted to the nodes in $\mathcal{D} \cup \mathcal{C}$ within time $2\mathbb{U}\tau$ according to Theorem 4.4.1. □

After time $2\mathbb{U}\tau$, all the data packets at $V \setminus (\mathcal{D} \cup \mathcal{C})$ will be collected to $\mathcal{D} \cup \mathcal{C}$ according to Corollary 4.4.1. Subsequently, we investigate the time consumption to collect all the data packets at $(\mathcal{D} \cup \mathcal{C}) \setminus \{s_0\}$ to the sink s_0.

Lemma 4.4.3 [7] *Assume that* $\mathcal{X}$ *is a disk of radius* r_d *and* $\mathcal{M}$ *is a set of points with mutual distance of at least* 1. *Then* $|\mathcal{X} \cap \mathcal{M}| \leq \frac{2\pi r_d^2}{\sqrt{3}} + \pi r_d + 1$.

Let $\kappa = \sqrt[\alpha]{c_2(2^{\mathcal{R}_0/W} - 1)} + 1$. It follows that $\mathcal{R}_0$-PCR $= \kappa \cdot r$. Then, we can obtain the following lemma by applying Lemma 4.4.3.

Lemma 4.4.4 *Assume that $\mathcal{X}$ is a disk of radius $\mathcal{R}_0$-PCR, then $|\mathcal{X} \cap (\mathcal{D} \cup \mathcal{C})| \leq \beta_\kappa + \beta_{\kappa+1}$, where $\beta_x = \frac{2\pi x^2}{\sqrt{3}} + \pi x + 1$, i.e. the number of dominators and connectors within the CR of a node is at most $\beta_\kappa + \beta_{\kappa+1}$ in DDC.*

Proof: Since $\mathcal{X}$ is a disk of radius $\mathcal{R}_0$-PCR, it is possible for some connectors in $\mathcal{X}$ only connecting some dominators out of disk $\mathcal{X}$ as shown in Figure 4.4. On the other hand, all the dominators adjacent to the connectors in $\mathcal{X} \cap \mathcal{C}$ must locate in a concentric disk of $\mathcal{X}$ with radius $\mathcal{R}_0$-PCR$+r = (\kappa + 1)r$, denoted by $\mathcal{X}'$ as shown in Figure 4.4.

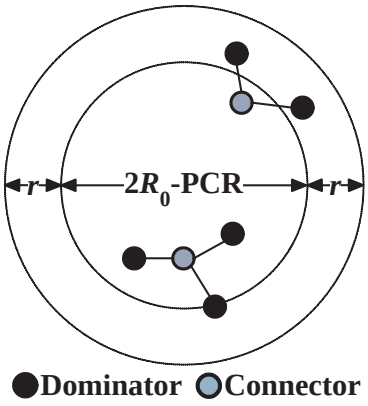


Figure 4.4 The number of dominators and connectors within the CR of a node.

Now, if r is normalized to 1, then $\mathcal{X}$ (respectively, $\mathcal{X}'$) is a disk of radius κ (respectively, $\kappa+1$), and $\mathcal{D}$ is a set of nodes with mutual distance of at least 1. Then, by Lemma 4.4.3, we have $|\mathcal{X} \cap \mathcal{D}| \leq \beta_\kappa = \frac{2\pi\kappa^2}{\sqrt{3}} + \pi\kappa + 1$ (respectively, $|\mathcal{X}' \cap \mathcal{D}| \leq \beta_{\kappa+1} = \frac{2\pi(\kappa+1)^2}{\sqrt{3}} + \pi(\kappa+1) + 1$), i.e. the number of the dominators within $\mathcal{X}$ (respectively, $\mathcal{X}'$) is at most β_κ (respectively, $\beta_{\kappa+1}$). Additionally, according to the aforementioned discussion and the CDS-based data collection tree construction process, each connector in $\mathcal{X} \cap \mathcal{C}$ must have a dominator parent located at disk $\mathcal{X}'$, which implies $|\mathcal{X} \cap \mathcal{C}| \leq |\mathcal{X}' \cap \mathcal{D}| \leq \beta_{\kappa+1}$. It follows that $|\mathcal{X} \cap (\mathcal{D} \cup \mathcal{C})| \leq \beta_\kappa + \beta_{\kappa+1}$ is proven. □

From Lemma 4.4.4, we can obtain the following corollary.

Corollary 4.4.2 *After time $2\mathbb{U}\tau$, every node in $(\mathcal{D} \cup \mathcal{C}) \setminus \{s_0\}$ with data packets for transmission can transmit at least one data packet to its parent node within time $2(\beta_\kappa + \beta_{\kappa+1})\tau$*

in DDC.

Proof: According to Lemma 4.4.4, there are at most $\beta_\kappa+\beta_{\kappa+1}$ dominators and connectors within the CR of a node. Furthermore, all the nodes in $V \setminus (\mathcal{D} \cup \mathcal{C})$ have no data packets for transmission after time $2\mathbb{U}\tau$ according to Corollary 4.4.1. Then, by the same technique used to prove Theorem 4.4.1, the conclusion of this corollary can be obtained. □

Based on Lemma 4.4.4 and Corollary 4.4.2, we can obtain the time consumption to collect all the data packets at $(\mathcal{D} \cup \mathcal{C}) \setminus \{s_0\}$ to the sink s_0 as shown in Theorem 4.4.2.

Theorem 4.4.2 *After time $2\mathbb{U}\tau$, it takes at most $2(n-\Delta_0)\cdot(\beta_\kappa+\beta_{\kappa+1})\cdot\tau$ time to collect all the data packets at $(\mathcal{D} \cup \mathcal{C}) \setminus \{s_0\}$ to the sink s_0 in DDC, where Δ_0 is the degree of s_0 in the data collection tree T.*

Proof: As shown in Corollary 4.4.1, after time $2\mathbb{U}\tau$, all the nodes in $V \setminus (\mathcal{D} \cup \mathcal{C})$ have no data packets for transmission, and meanwhile, s_0 has received at least Δ_0 data packets according to Theorem 4.4.1, since it has Δ_0 child nodes in T. Subsequently, s_0 receives at least one data packet in every $2(\beta_\kappa+\beta_{\kappa+1})\tau$ time according to Corollary 4.4.2. Thus, it takes at most $(n-\Delta_0)\cdot 2(\beta_\kappa+\beta_{\kappa+1})\tau$ time to collect all the data packets at $(\mathcal{D} \cup \mathcal{C}) \setminus \{s_0\}$ to the sink s_0 after time $2\mathbb{U}\tau$. □

Theorem 4.4.3 *The lower bound of data collection capacity achieved by DDC is $\Omega(\frac{1}{2(\beta_\kappa+\beta_{\kappa+1})}\cdot W)$, which is scalable and order-optimal.*

Proof: According to Theorem 4.4.1 and Theorem 4.4.2, to collect all the n data packets to the sink, the time consumption

$$\mathcal{T} \le 2\mathbb{U}\tau + 2(n-\Delta_0)\cdot(\beta_\kappa+\beta_{\kappa+1})\cdot\tau \tag{4.32}$$

$$= [(2\log n + \frac{\pi R^2(e^2-1)}{c_1}) + 2(n-\Delta_0)\cdot(\beta_\kappa+\beta_{\kappa+1})]\cdot\tau \tag{4.33}$$

$$\le [2\log n + \frac{\pi R^2(e^2-1)}{c_1} + 2(\beta_\kappa+\beta_{\kappa+1})n]\cdot\tau \tag{4.34}$$

$$= O(2(\beta_\kappa+\beta_{\kappa+1})n\cdot\tau). \tag{4.35}$$

Thus, the achievable data collection capacity of DDC is $\mathbb{C} = \frac{nB}{\mathcal{T}} \geq \frac{nB}{O(2(\beta_\kappa+\beta_{\kappa+1})n\cdot\tau)} = \Omega(\frac{1}{2(\beta_\kappa+\beta_{\kappa+1})}\cdot W)$. As mentioned before, the upper bound capacity of data collection is $O(W)$ [4][5], and $\beta_{\kappa+1}$ is a constant value depending on $\mathcal{R}_0$, which implies the achievable data collection capacity of the DDC algorithm is order-optimal. Furthermore, since $\mathbb{C}$ is independent of network size n, DDC is scalable. □

4.5 $\mathcal{R}_0$-PCR-based Distributed Data Aggregation

As introduced in Part 2, data gathering can be categorized as data collection and data aggregation. Therefore, for completeness, we in this section discuss how to apply the derived proper carrier-sensing range $\mathcal{R}_0$-PCR to distributed data aggregation in WSNs.

In data aggregation, multiple data packets can be aggregated into one data packet by applying an aggregation function, e.g. MAX, MIN, SUM, etc. Formally, the data aggregation problem can be defined as follows. Let $X, Y \subseteq V$ and $X \cap Y = \emptyset$. The data of the nodes in X is said to be *aggregated* to the nodes in Y in a time slot, if all the nodes in X can transmit their data packets to the nodes in Y concurrently and interference-freely during a time slot. Here, X is called a *transmitter set*. Then, the data aggregation problem can be defined as to seek a *data aggregation schedule* which consists of a sequence of transmitter sets $X_1, X_2, \cdots, X_M$, such that

1. $\forall 1 \leq i \neq j \leq M$, $X_i \cap X_j = \emptyset$;
2. $\bigcup_1^M X_i = V \setminus \{s_0\}$, where M is the latency of this data aggregation schedule;
3. Data can be aggregated from X_i to $V \setminus \bigcup_{j=1}^{i} X_j$ during time slot i for $i = 1, 2, \cdots, M$.

Ever since the data aggregation problem is raised, extensive research has been conducted on this issue ([7],[60]-[65], and references therein), especially for the Minimum-Latency Aggregation Schedule (MLAS) problem, which tries to obtain a data aggregation schedule with the objective to minimize the latency (minimize M). In [60], [61] and [7], several centralized data aggregation algorithms are proposed under the Unit Disk Graph (UDG) model

and the protocol interference model. Chen et al. [60] proved that the MLAS problem is NP-hard. Furthermore, they designed a $(\Delta - 1)$-approximation algorithm for this problem, where Δ is the maximum degree of the topological graph of a network. Subsequently, Huang et al. [61] proposed another data aggregation algorithm which has a better performance. By analysis, they showed that the delay of their algorithm is upper bounded by $23\mathfrak{R} + \Delta - 18$ ($\mathfrak{R} \sim L$ and L is defined in Section 4.4), where $\mathfrak{R}$ is the network radius. Recently, Wan et al. [7] proposed three data aggregation algorithms of latency upper bounded by $15\mathfrak{R} + \Delta - 4$, $2\mathfrak{R} + O(\log \mathfrak{R}) + \Delta$, and $(1 + O(\log \mathfrak{R}/\sqrt[3]{\mathfrak{R}}))\mathfrak{R}$, respectively. Xu et al. [62] studied periodic query scheduling for data aggregation with the minimum delay consideration. They designed the centralized aggregation scheduling algorithms under various wireless interference models, and analyzed the induced delay of each algorithm. As explained in Section 4.1, centralized algorithms have many shortcomings in distributed wireless networks. To overcome these shortcomings, some state-of-the-art distributed algorithms are proposed under the UDG model and the protocol interference model [63][64][65]. In [63], Yu et al. proposed a distributed CDS-based data aggregation schedule algorithm with latency upper bounded by $24\mathfrak{D} + 6\Delta + 16$, where $\mathfrak{D}$ is the network diameter. Xu et al. [64] also proposed a distributed data aggregation algorithm with a better latency bound of $16\mathfrak{R}' + 6\Delta - 14$, where $\mathfrak{R}'$ is the inferior network radius which satisfies $\mathfrak{R}' \leq \mathfrak{R} \leq \mathfrak{D} \leq 2\mathfrak{R}'$. The most recently published distributed data aggregation algorithm is [65], in which Li et al. proposed an aggregation scheme of latency upper bounded by $16\mathfrak{R}' + \Delta - 14$.

Unlike the previous works, we design an $\mathcal{R}_0$-PCR-based *Distributed Data Aggregation* (DDA) algorithm. The main differences between this DDA and the previous works can be summarized as follows. First, DDA is a distributed and asynchronous algorithm while many previous algorithms (e.g. [7], [60]-[62]) are centralized. Since WSNs tend to be distributed systems, distributed and asynchronous algorithms are more practical and suitable. Second, DDA runs under the generalized physical interference model while most of the previous works are under the UDG model or the protocol interference model. Compared with the generalized physical interference model, the protocol interference model is simplified and

can make the analysis process much easier. On the other hand, the generalized physical interference model considers the aggregated interference in a WSN, which is more practical as well as more complicated.

The description of our algorithm is shown in Algorithm 4. DDA is similar to DDC. The main difference is that each node s_i $(1 \leq i \leq n)$ only transmits one data packet to its parent node, while in DDC, it may have to transmit multiple data packets to its parent node, i.e. the traffic load of a data collection task is much heavier than that of a data aggregation task.

Algorithm 4: The DDA Algorithm

input : CDS-based data collection tree T, $\mathcal{R}_0$-PCR
output: a distributed asynchronous data aggregation plan

```
1  s_i (1 ≤ i ≤ n) sets its CR as R_0-PCR according to the required threshold of data
   receiving rate R_0;
2  while s_0 has not received the aggregation data do
3      if s_i is a leaf node in T or s_i has received the aggregation data from all of its
       children in T then
4          if s_i is a non-leaf node then
5              s_i obtains the aggregation value of its data and the data of its children by
               applying the aggregation function;
6          s_i randomly sets a backoff time t_i for its data transmission in window (0, τ_w];
7          while t_i is not countdown to 0 do
8              s_i senses the channel with R_0-PCR;
9              if s_i senses that the channel is busy then
10                 s_i stops the countdown process (the backoff timer is frozen) until the
                   channel becomes free again;
11             if s_i senses that the channel is free then
12                 t_i − −;
13         if t_i == 0, i.e. the backoff timer expires then
14             s_i transmits the aggregation data to its parent node in T.
```

In Algorithm 4, the routing structure is a CDS-based data collection tree T as in DDC, and we also assume no two transmitters within the CR of each other have their backoff timers expired at the exactly same time instant.

Now, we analyze the delay performance of DDA. Similar to the delay of DDC, we can

obtain the upper bound of the time consumption of DDA as shown in Theorem 4.5.1.

Theorem 4.5.1 *The induced delay of DDA is upper bounded by* $\log n + (\beta_\kappa + \beta_{\kappa+1} - 1)L + c_3$ *time slots, where* L *is the hight of the data collection (aggregation) tree* T*, and* $c_3 = \frac{\pi R^2(e^2-1)}{2c_1} - \beta_\kappa - \beta_{\kappa+1}$ *is a constant value depending on* $\mathcal{R}_0$*-PCR.*

Proof: From Lemma 4.4.2, the upper bound of the number of nodes within a disk of radius $\mathcal{R}_0$-PCR is $\mathbb{U} = \log n + \frac{\pi R^2(e^2-1)}{2c_1}$. Therefore, for any node, it waits at most $\mathbb{U} - 2$ time slots before transmitting its data to its parent node (minus two means the transmitter and its parent node are not counted). Therefore, it takes at most $(\mathbb{U} - 1) \cdot \tau$ time to aggregate all the data at $V \setminus (\mathcal{C} \cup \mathcal{D})$ to $\mathcal{C} \cup \mathcal{D}$ according to the schedule strategy in DDA. After $(\mathbb{U} - 1) \cdot \tau$ time, there is no data for transmission at nodes in $V \setminus (\mathcal{C} \cup \mathcal{D})$. Based on Lemma 4.4.4, the number of dominators and connectors within a disk of radius $\mathcal{R}_0$-PCR is upper bounded by $\beta_\kappa + \beta_{\kappa+1}$. Consequently, according to DDA, a node in $\mathcal{C} \cup \mathcal{D}$ has an opportunity to transmit one data packet within time $(\beta_\kappa + \beta_{\kappa+1} - 1) \cdot \tau$. Considering the hight of the data collection (aggregation) tree T is L (which implies the number of hops from the sink to any node in $\mathcal{C} \cup \mathcal{D}$ is at most $L - 1$), the number of time slots consumed by DDA is upper bounded by

$$(\mathbb{U} - 1) + (\beta_\kappa + \beta_{\kappa+1} - 1)(L - 1) \tag{4.36}$$

$$= \mathbb{U} + (\beta_\kappa + \beta_{\kappa+1} - 1)L - \beta_\kappa - \beta_{\kappa+1} \tag{4.37}$$

$$= \log n + (\beta_\kappa + \beta_{\kappa+1} - 1)L + \frac{\pi R^2(e^2-1)}{2c_1} - \beta_\kappa - \beta_{\kappa+1} \tag{4.38}$$

$$= \log n + (\beta_\kappa + \beta_{\kappa+1} - 1)L + c_3, \tag{4.39}$$

where $c_3 = \frac{\pi R^2(e^2-1)}{2c_1} - \beta_\kappa - \beta_{\kappa+1}$. □

4.6 Data Collection and Aggregation under the Poisson Distribution Model

In Section 4.2, we assume that all the sensor nodes are independent and identically distributed. Based on that network distribution model, we obtain the achievable capacity of the proposed data collection method DDC, which is order-optimal, and the delay upper

bound of the designed data aggregation method DDA. To be more general, in this section, we consider another frequently employed non-i.i.d. model, named the *Poisson distribution* model, and analyze the performances of DDC and DDA.

Under the Poisson distribution model, we assume that one sink node s_0 and n sensor nodes $s_1, s_2, \ldots, s_n$ are distributed according to a two-dimensional Poisson point process with density λ in some area with size $A = c_1 n$. To make data collection and aggregation meaningful, we also assume that the network is connected. Then, by the same method in Section 4.4, a CDS-based data collection tree T can be constructed. Therefore, we can still exploit DDC and DDA to finish data collection and aggregation tasks under the Poisson distribution model. Now, we analyze the delay performance of DDC and DDA.

Let $R = \mathcal{R}_0$-PCR $= (\sqrt[\alpha]{c_2(2^{\mathcal{R}_0/W} - 1)} + 1) \cdot r = \kappa \cdot r$. We first analyze the average/upper bound of the number of sensor nodes within the CR of a node as shown in the following lemma.

Lemma 4.6.1 *Let the random variable X denote the number of sensor nodes within the CR of a node. Then, we have*

(i) $\mathbf{E}[X] = \pi\lambda R^2$;

(ii) it is almost sure that the number of sensor nodes within the CR of a node is upper bounded by $a \log n$, *where* $a = \arg\min_{\epsilon>0} (\frac{2}{\epsilon} + \frac{\pi\lambda R^2(e^\epsilon - 1)}{\epsilon \log n})$.

Proof: (i) Since the sensor nodes are distributed according to a two-dimensional Poisson point process with density λ, we have

$$\mathbf{E}[X] = \sum_{k=1}^{+\infty} \Pr(X = k) \cdot k \tag{4.40}$$

$$= \sum_{k=1}^{+\infty} \frac{(\pi\lambda R^2)^k}{k!} \exp(-\pi\lambda R^2) \cdot k \tag{4.41}$$

$$= \pi\lambda R^2. \tag{4.42}$$

(ii) Similar to the proof in Lemma 4.4.2, applying the Chernoff bound and for any $\epsilon > 0$,

we have

$$\Pr[X > a\log n] \leq \Pr[X \geq a\log n] \tag{4.43}$$

$$\leq \min_{\epsilon>0} \frac{\mathbf{E}[e^{\epsilon X}]}{e^{\epsilon a\log n}} = \min_{\epsilon>0} \frac{\exp(\pi\lambda R^2(e^{\epsilon}-1))}{e^{\epsilon a\log n}} \tag{4.44}$$

$$= \min_{\epsilon>0} \exp(\pi\lambda R^2(e^{\epsilon}-1) - \epsilon a\log n) \tag{4.45}$$

$$= \exp(-2\log n) \leq \frac{1}{n^2}. \tag{4.46}$$

Since $\sum_{n>0} \frac{1}{n^2}$ is upper bounded by $\frac{\pi^2}{6}$, it follows that the number of sensor nodes within the CR of a node is upper bounded by $a\log n$ almost surely, where $a = \arg\min_{\epsilon>0}(\frac{2}{\epsilon} + \frac{\pi\lambda R^2(e^{\epsilon}-1)}{\epsilon\log n})$. □

Based on Lemma 4.6.1, it is reasonable to take $a\log n$ as the upper bound of the number of sensor nodes within the CR of a node. Then, we have the following theorem, which indicates the upper bound of the induced delay of our data collection algorithm DDC under the Poisson distribution model.

Theorem 4.6.1 *Under the Poisson distribution model, the induced delay of DDC to collect all the data (n data packets) to the sink is upper bounded by* $2(a\log n+(n-\Delta_0)(\beta_\kappa+\beta_{\kappa+1}))\cdot\tau$, *where* $a = \arg\min_{\epsilon>0}(\frac{2}{\epsilon} + \frac{\pi\lambda R^2(e^{\epsilon}-1)}{\epsilon\log n})$.

Proof: Based on Lemma 4.6.1 and by similar methods to Theorem 4.4.1 and Corollary 4.4.1, it can be proven that the time consumption to collect all the data packets at $V\setminus(\mathcal{D}\cup\mathcal{C})$ to $\mathcal{D}\cup\mathcal{C}$ is upper bounded by $2a\log n\tau$. Subsequently, similar to Theorem 4.4.2, the time consumption to collect all the n data packets to the sink is

$$\mathcal{T} \leq 2a\log n\tau + 2(n-\Delta_0)(\beta_\kappa+\beta_{\kappa+1})\tau \tag{4.47}$$

$$= 2(a\log n + (n-\Delta_0)(\beta_\kappa+\beta_{\kappa+1}))\cdot\tau, \tag{4.48}$$

where $a = \arg\min_{\epsilon>0}(\frac{2}{\epsilon} + \frac{\pi\lambda R^2(e^{\epsilon}-1)}{\epsilon\log n})$. □

Based on Theorem 4.6.1, the achievable data collection capacity of DDC can be obtained

as shown in the following theorem.

Theorem 4.6.2 *Under the Poisson distribution model, the achievable data collection capacity of DDC is lower bounded by* $\Omega(\frac{1}{2(\beta_\kappa+\beta_{\kappa+1})} \cdot W)$*, which is scalable and order-optimal.*

Proof: By a similar method to Theorem 4.4.3, this theorem can be proven. □

Now, we analyze the induced delay of DDA under the Poisson distribution model, which is shown in Theorem 4.6.3

Theorem 4.6.3 *Under the Poisson distribution model, the induced delay of DDA is upper bounded by* $a \log n + (\beta_\kappa + \beta_{\kappa+1} - 1)L - c_4$*, where* $a = \arg\min_{\epsilon>0} (\frac{2}{\epsilon} + \frac{\pi\lambda R^2(e^\epsilon - 1)}{\epsilon \log n})$ *and* $c_4 = \beta_\kappa + \beta_{\kappa+1}$ *are constant values.*

Proof: By a similar method to Theorem 4.5.1, this theorem can be proven. □

4.7 Simulation Results

In this section, we present simulation results to validate the performances of DDC and DDA. In all the simulations, we consider the WSNs consisting of one sink node and n sensor nodes which are randomly deployed in a square area with size $A = c_1 n$. Thus the node density is $\frac{1}{c_1}$. Since our primary concern is the achievable capacity and scalability (respectively, induced delay) of DDC (respectively, DDA), we make some simplification and normalization on the simulation settings. The maximum transmission radius of a node is normalized to one and any node can work on the Re-Start (RS) mode with the IPCS technique [80]. During the data collection period, every node produces a data packet whose size is also normalized to one. Furthermore, all the nodes work with the same transmission power $P = 1$ and over a common wireless channel with bandwidth normalized to one, which implies the transmission time of a data packet τ is 1 in the ideal case. Then, we set the backoff contention window $\tau_w = \frac{1}{10}$ for DDC and DDA in all the simulations. For a data transmission, the background noise is negligible compared with the interference brought by concurrent transmissions. Hence, we do not consider the background noise. For other

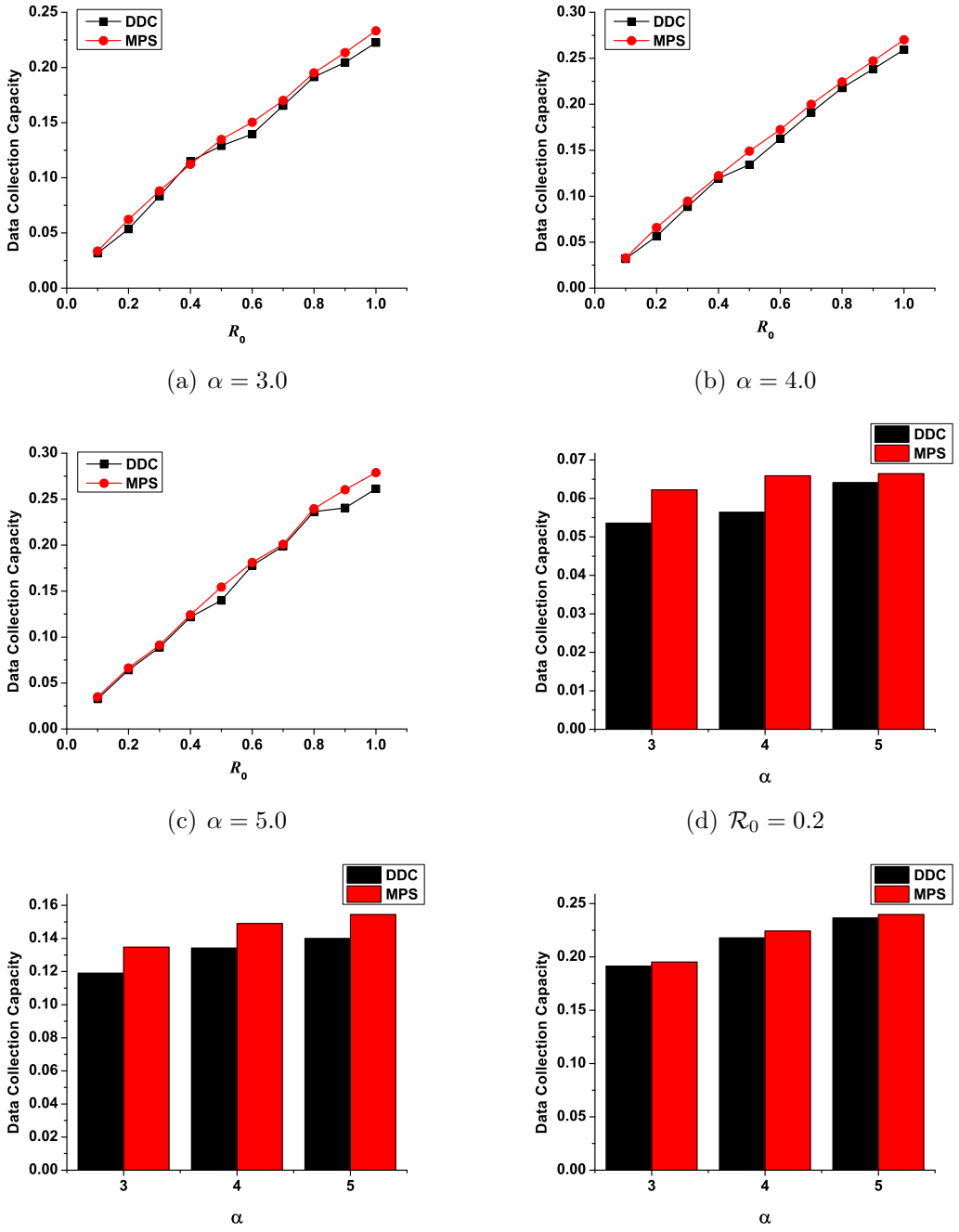
important system parameters, e.g. the network size A, the node density $\frac{1}{c_1}$, the number of nodes n, the path loss exponent α, etc., we specify them later in each group of simulations.

The compared algorithm for DDC is the *Multi-Path Scheduling* (MPS) algorithm proposed in [5], which is the most recently published centralized and synchronized data collection method under the simplified *protocol interference model* for WSNs. In MPS, the interference radius $R_I = \eta \cdot r(\eta \geq 1)$, where η is a constant and r is the communication radius of a node. Thus, in the following simulations, we set $R_I = \mathcal{R}_0$-PCR, which guarantees that MPS can also initiate data transmissions with a satisfied data receiving rate $\mathcal{R}_0$. The compared algorithm for DDA is the *Enhanced Pipelined Aggregation Scheduling* (E-PAS) algorithm [7], which is the best and latest centralized data aggregation algorithm. Since E-PAS is also designed under the protocol interference model, we set the interference radius of E-PAS to $\mathcal{R}_0$-PCR according to different $\mathcal{R}_0$ values. In the following, each group of simulations is repeated for 100 times and the results are the average values.

4.7.1 DDC Capacity *vs.* $\mathcal{R}_0$ and α

In this subsection, we consider the WSNs deployed in a square area with size $A = 20 \times 20$ and the node density is 3. The impacts of $\mathcal{R}_0$ and α on the capacities of DDC and MPS are shown in Figure 4.5. From Figure 4.5(a)-(c), we can see that with the increase of $\mathcal{R}_0$, the achievable capacities of both DDC and MPS increase. Although a large $\mathcal{R}_0$ implies a large $\mathcal{R}_0$-PCR (shown in Figure 4.2), which further implies that fewer nodes can conduct transmissions concurrently, on the other hand a large $\mathcal{R}_0$-PCR also implies short transmission time of a data packet. Furthermore, with the increase of $\mathcal{R}_0$, the decrease of the transmission time of a data packet is faster than the increase of $\mathcal{R}_0$-PCR, i.e. $\mathcal{R}_0$ dominates the achievable data collection capacity. It follows that a large $\mathcal{R}_0$ leads to a high capacity for both DDC and MPS.

From Figure 4.5(d)-(f), we can see that with the increase of α, the achievable capacities of DDC and MPS also increase. This is because, for any transmission, the interference impact from other concurrent transmissions decreases quickly with the increase of α. Thus, a large



(a) $\alpha = 3.0$

(b) $\alpha = 4.0$

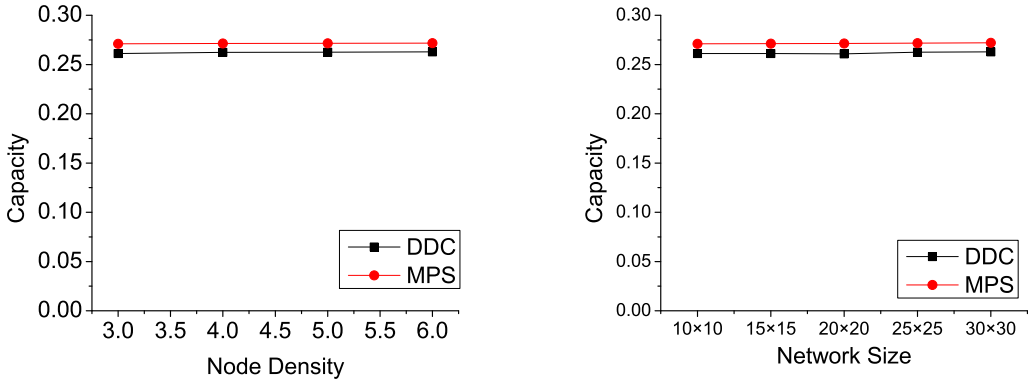
(c) $\alpha = 5.0$

(d) $\mathcal{R}_0 = 0.2$

(e) $\mathcal{R}_0 = 0.5$

(f) $\mathcal{R}_0 = 0.8$

Figure 4.5 DDC capacity *vs.* MPS capacity.



(a) (b)

Figure 4.6 DDC/MPS capacity *vs.* Node density/Network size.

α implies a small $\mathcal{R}_0$-PCR and results in more nodes being able to initiate transmissions concurrently. Therefore, the achievable data collection capacities of DDC and MPS increase when α increases.

From Figure 4.5, we can also see that DDC achieves similar data collection capacity to the centralized and synchronous MPS, although DDC is a distributed and asynchronous data collection algorithm. The reason is that we set a proper CR for DDC. By setting the CR of each node as $\mathcal{R}_0$-PCR, as many as possible nodes can initiate data transmissions concurrently with a guaranteed data receiving rate at the receivers. This can also be seen from Theorem 4.3.1. From the proof of Theorem 4.3.1, by packing all the possible concurrent data transmissions in the densest manner, we obtain a small proper CR maximizing the number of concurrent transmissions. Consequently, as many as possible transmissions can be scheduled simultaneously without interference at any time, inducing high achievable data collection capacity of DDC. Particularly, the average capacity differences between DDC and MPS are 5.25%, 4.99%, and 4.27% when $\alpha = 3$, $\alpha = 4$, and $\alpha = 5$, respectively, which indicates that DDC achieves comparable capacity as centralized and synchronized MPS.

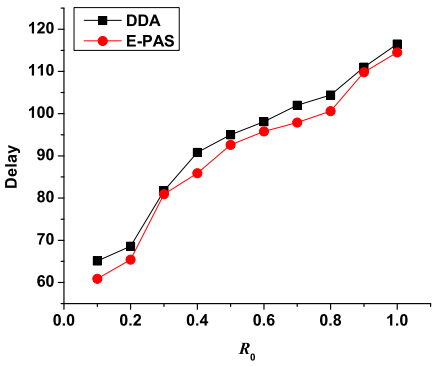
4.7.2 Scalability of DDC

We examine the scalability of DDC with respect to the number of sensor nodes in a WSN. In the following simulations, we set the path loss exponent α to 4, $\mathcal{R}_0$ to 1 (i.e. the CR for DDC is 1-PCR), the default network size to 10×10, and the default node density to be 4. The impacts of the node density and the network size on the scalability and achievable capacities of DDC and MPS are shown in Figure 4.6. where we can see that with the increase of the number of sensor nodes (by fixing the network size and increasing the node density in Figure 4.6(a) and fixing the node density and increasing the network size in Figure 4.6(b)), the acheivable capacity of DDC keeps stable as that of centralized and synchronized MPS, which implies DDC is scalable with respect to n, the number of sensor nodes in a WSN. This is because the capacity of DDC only depends on $\mathcal{R}_0$-PCR, which is a distance-dependent parameter. Thus, DDC is scalable for WSNs with different network sizes and node densities.

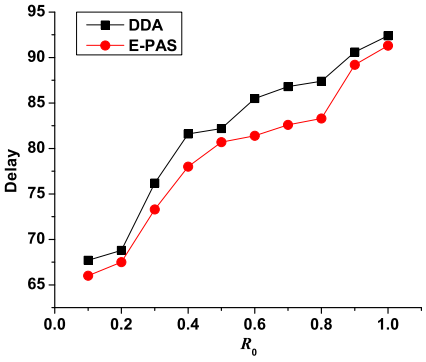
4.7.3 Performance of DDA

In this subsection, we examine the performance of DDA with respect to α, $\mathcal{R}_0$, and the number of sensor nodes n. In all the simulations, we set the node density to 4. The results are shown in Figure 4.7.

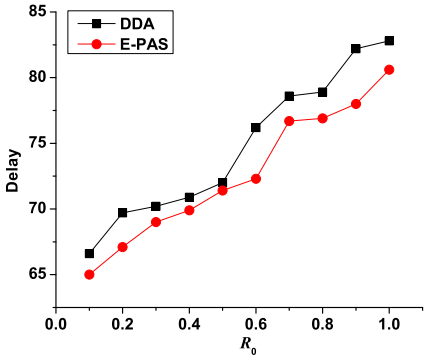
From Figure 4.7(a)-(c), we can see that with the increase of the guaranteed data receiving rate $\mathcal{R}_0$, the induced delay by both DDA and E-PAS increases for different α values. This is different from the data collection situation, where the capacities of both DDC and MPS increase when $\mathcal{R}_0$ increases. This is because: (i) with the increase of $\mathcal{R}_0$, the corresponding $\mathcal{R}_0$-PCR increases as well (which can be seen from Figure 4.2). It follows that fewer data transmissions can be conducted simultaneously in DDA and E-PAS. On the other hand, even a larger $\mathcal{R}_0$ implies more data can be transmitted during one data transmission, i.e. fewer transmission times. The induced delay of DDA and E-PAS still increases with the increase of $\mathcal{R}_0$ since $\mathcal{R}_0$-PCR now plays the dominating role in data aggregation; (ii) data collection has much more traffic (which is of order of $O(n^2)$) than data aggregation (which is of order of $O(n)$). Therefore, the data transmission rate (decided by $\mathcal{R}_0$) has more impacts



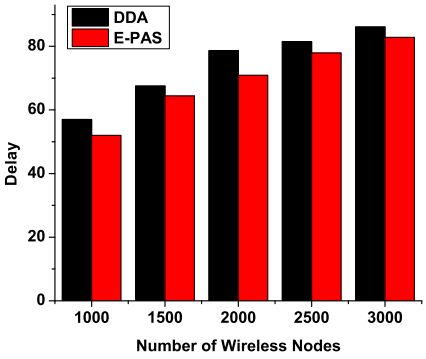
(a) $\alpha = 3$, $n = 2000$



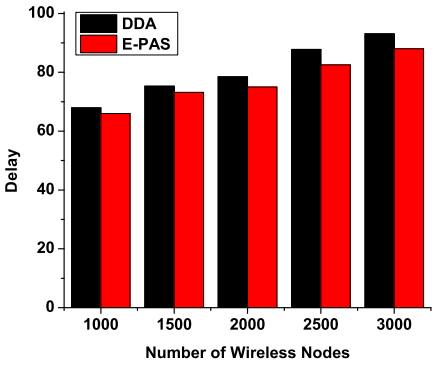
(b) $\alpha = 4$, $n = 2000$



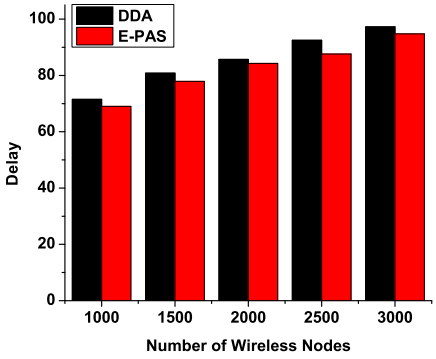
(c) $\alpha = 5$, $n = 2000$



(d) $\mathcal{R}_0 = 0.2, \alpha = 4$



(e) $\mathcal{R}_0 = 0.5, \alpha = 4$



(f) $\mathcal{R}_0 = 0.8, \alpha = 4$

Figure 4.7 Data aggregation delay of DDA and E-PAS.

on the delay (as well as capacity) of data collection, while the data transmission concurrency (decided by $\mathcal{R}_0$-PCR) has more impacts on the delay of data aggregation, i.e., the guaranteed data receiving rate ($\mathcal{R}_0$) will dominate the delay increasing of data collection while the carrier-sensing (interference) range ($\mathcal{R}_0$-PCR) will dominate the delay increasing of data aggregation. From Figure 4.7(a)-(c), we can also see that DDA has similar delay performance to E-PAS although DDA schedules data transmission in a distributed and asynchronous manner. On average, the delay differences between DDA and E-PAS in Figure 4.7(a)-(c) are around 3.1%, 3.2%, and 2.6% respectively, which are quite small.

The data aggregation delay of DDA and E-PAS in WSNs with different sizes is shown in Figure 4.7(d)-(f). From Figure 4.7(d)-(f), we can see that the induced delay of DDA and E-PAS increases when the network becomes larger. The reason is straightforward since more sensor nodes imply heavier traffic load. From Figure 4.7(d)-(f), we can also see that the delay difference between DDA and E-PAS is very small. Particularly, in Figure 4.7(d)-(f), the average delay differences between DDA and E-PAS are about 6.1%, 4.4%, and 3.3% respectively, which implies DDA has comparable delay performance as the best centralized data aggregation algorithm.

4.8 Conclusion

Since WSNs in practice tend to be distributed asynchronous systems and most of the existing works study the network capacity issues for centralized synchronized WSNs, we investigate the achievable data collection capacity for distributed asynchronous WSNs in this part. To avoid data transmission collisions/interferences, we derive an $\mathcal{R}_0$-Proper Carrier-sensing Range ($\mathcal{R}_0$-PCR) under the generalized physical interference model. By taking $\mathcal{R}_0$-PCR as its carrier-sensing range, any node can initiate a data transmission with a guaranteed data receiving rate. Subsequently, based on the obtained $\mathcal{R}_0$-PCR, we propose a scalable Distributed Data Collection (DDC) algorithm with fairness consideration for asynchronous WSNs. Theoretical analysis of DDC surprisingly shows that its achievable data collection capacity is also order-optimal as that of centralized synchronized algorithms. Moreover, we

study how to apply $\mathcal{R}_0$-PCR to distributed data aggregation in asynchronous WSNs, and propose a Distributed Data Aggregation (DDA) algorithm. By analysis, the delay bound of DDA is present. To be more general, we investigate the delay and capacity of DDC and DDA under the Poisson node distribution model. The analysis again shows that DDC is order-optimal and scalable with respect to achievable data collection capacity. The extensive simulation results demonstrate that DDC has comparable data collection capacity compared with the most recently published centralized and synchronized data collection algorithm, and DDC is scalable in WSNs with different network sizes and node densities. DDA also has similar performance to the latest and best centralized data aggregation algorithm.

The future work can be conducted along the following directions: first, we would like to apply the derived PCR to other issues in WSNs, e.g. broadcast scheduling, multicast scheduling, etc, and propose efficient distributed solutions for these issues. Second, we study the data collection and aggregation problems for randomly deployed WSNs in this part. However, it is still an open problem to design an order-optimal data collection algorithm in arbitrarily distributed WSNs. The reason is that the nodes may distribute according to any model in arbitrary WSNs, and thus there are many challenges to design an order-optimal data collection algorithm with accurate capacity analysis. Therefore, we will study order-optimal distributed data collection and aggregation issues for arbitrarily distributed WSNs. Finally, there is a trade-off between network capacity and lifetime. In this work, we focus on designing a distributed data collection algorithm with the objective to maximize the achievable capacity. In the future work, we would like to study how to implement an order-optimal data collection algorithm and meanwhile maximize network lifetime.

PART 5

CONTINUOUS DATA AGGREGATION AND CAPACITY IN PROBABILISTIC WIRELESS SENSOR NETWORKS

5.1 Introduction

For completeness, we study the snapshot and continuous data aggregation problems for probabilistic WSNs in this part. In data gathering WSNs, the problem of collecting the aggregated value of one snapshot is called *Snapshot Data Aggregation* (SDA). The problem of collecting the aggregated value of each snapshot of multiple continuous snapshots is called *Continuous Data Aggregation* (CDA). For snapshot data aggregation and continuous data aggregation, we use the ratio between the amount of data been aggregated and the time used to transmit the aggregated values of these data to the sink, referred to as *snapshot data aggregation capacity* and *continuous data aggregation capacity* respectively, to measure their achievable network capacity[1].

As discussed in Part 3, most of the existing works that study the network capacity issue are based on the ideal *Deterministic Network Model* (DNM), where any pair of nodes in a network is either *connected* or *disconnected*. If two nodes are connected, i.e. there is a deterministic link between them, then a successful data transmission can be guaranteed as long as there is no collision. Otherwise, if two nodes are disconnected, the direct communication between them is assumed to be impossible. However, in real applications, this deterministic network model assumption is too ideal and not practical due to the "*transitional region phenomenon*" [74][75]. With the transitional region phenomenon, a large number of network links (probably more than 90% [74]) become unreliable, named *lossy links* [74]. Even without collisions, data transmission over a lossy link is successfully conducted with a certain proba-

[1]Without confusion, we use snapshot data aggregation/continuous data aggregation capacity and network capacity interchangeably in the following of this part.

bility, rather than being completely guaranteed. Therefore, a more practical network model for WSNs is the *Probabilistic Network Model* (PNM) [74], in which data communication over a link is successful with a certain probability rather than always being successful or always fail.

As mentioned before, for the network capacity issues (including uni/multi/broad-cast, data collection/aggregation capacities), most of the existing works are based on the ideal DNM rather than the more practical PNM. This motivates us to study the achievable network capacity of WSNs under the realistic probabilistic network model, i.e. for probabilistic WSNs. Specifically, in this part, we investigate the achievable network capacities of snapshot data aggregation and continuous data aggregation under the probabilistic network model. When studying the snapshot data aggregation and continuous data aggregation capacities, we first partition the network into *cells* and derive the lower and upper bounds of the number of sensors within each cell (as in Part 3). Afterwards, we use two vectors to further partition all the cells into different *equivalent color classes* (as the compatible transmission cell set in Part 3). Based the equivalent color classes, we design a *Cell-based Aggregation Scheduling* (CAS) algorithm for snapshot data aggregation, and a *Level-based Aggregation Scheduling* (LAS) algorithm for continuous data aggregation. Furthermore, we prove that both CAS and LAS are order-optimal by analyzing their achievable network capacities. Particularly, the main contributions of this part are summarized as follows:

1. Inspired by the network partition method in [78], we first partition a WSN into cells and use two vectors to further partition these cells into equivalent color classes. According to the obtained cells and equivalent color classes, we design a two-phase *Cell-based Aggregation Scheduling* (CAS) algorithm for the SDA problem in probabilistic WSNs. In the first phase, all the *non-local aggregation nodes* transmit their data packets to the *local aggregation node* in the same cell. In the second phase, all the local aggregation nodes transmit the local aggregation values along the constructed data aggregation tree to the sink. Theoretical analysis shows that the achievable capacities of CAS are all $\Omega(\frac{p_o\sqrt{en\log n}}{2\omega}\cdot W)$ in the worst case, in the average case, and in the best case, where

p_o is the promising transmission threshold probability (Section 5.2), n is the number of sensor nodes in the considering WSN, ω is a constant value, and W is the bandwidth of the wireless channel. Moreover, we study the upper bound capacity of the SDA problem, which is $O(\frac{p_o\sqrt{en\log n}}{3}\cdot W)$. This implies that CAS has successfully achieved order optimal capacities in all the cases.

2. We propose a *Level-based Aggregation Scheduling* (LAS) algorithm for the CDA problem in probabilistic WSNs. LAS gathers the aggregation values of continuous snapshots by forming a data aggregation/transmission pipeline on the *segments* and scheduling the *cell-levels* in a *cell-level class* concurrently. Theoretical analysis of LAS shows that its achievable network capacity is

$$\begin{cases} \Omega(\frac{\sqrt{e}p_oN}{13.4\omega}\sqrt{\frac{n}{\log n}}\cdot W), & \text{if } N = O(\sqrt{\frac{n}{\log n}}); \\ \Omega(\frac{p_o}{13.4\omega^2}\frac{n}{\log n}\cdot W), & \text{if } N = \Omega(\sqrt{\frac{n}{\log n}}). \end{cases}$$

in the worst case,

$$\begin{cases} \Omega(\frac{p_oN}{2\sqrt{e}\omega}\sqrt{\frac{n}{\log n}}\cdot W), & \text{if } N = O(\sqrt{\frac{n}{\log n}}); \\ \Omega(\frac{p_o}{2e\omega^2}\frac{n}{\log n}\cdot W), & \text{if } N = \Omega(\sqrt{\frac{n}{\log n}}). \end{cases}$$

in the average case, and

$$\begin{cases} \Omega(\frac{e\sqrt{e}p_oN}{\omega}\sqrt{\frac{n}{\log n}}\cdot W), & \text{if } N = O(\sqrt{\frac{n}{\log n}}); \\ \Omega(\frac{ep_o}{\omega^2}\frac{n}{\log n}\cdot W), & \text{if } N = \Omega(\sqrt{\frac{n}{\log n}}). \end{cases}$$

in the best case, where N is the number of snapshots in a continuous data aggregation task. We also investigate the upper bound capacity of the CDA problem, which is

$$\begin{cases} O(\frac{2e\sqrt{e}p_oN}{3}\sqrt{\frac{n}{\log n}}\cdot W), & \text{if } N = O(\sqrt{\frac{n}{\log n}}); \\ O(\frac{2ep_o}{9}\frac{n}{\log n}\cdot W), & \text{if } N = \Omega(\sqrt{\frac{n}{\log n}}). \end{cases}.$$

This implies that LAS has already achieved optimal capacities in order in every case.

3. To be more general, we further theoretically analyze the capacity performance of CAS and LAS under the *Poisson point distribution* model. The analysis show that CAS and LAS can also achieve order optimal capacities under the Poisson distribution model.

4. We also conduct extensive simulations to validate the performances of CAS and LAS in probabilistic WSNs. Evaluation results indicate that CAS and LAS can improve the SDA and CDA capacities, as well as network lifetime, of probabilistic WSNs significantly, compared with the latest SDA and CDA methods for deterministic WSNs, respectively.

The rest of this part is organized as follows: In Section 5.2, we give the PNM and make some assumptions. In Section 5.3, we discuss the network partition method, which is crucial for the following data aggregation scheduling algorithms. The Cell-based Aggregation Scheduling (CAS) algorithm for SDA is proposed and analyzed in Section 5.4. In Section 5.5, we design the Level-based Aggregation Scheduling (LAS) algorithm for CDA, and we also derive the achievable capacity of LAS theoretically. To make our work more general, we also analyze the capacity performance of CAS and LAS under the non-i.i.d. node distribution model in Section 5.6, which turns out to be order optimal either. In Section 5.7, the simulations are conducted to validate the performances of CAS and LAS, and we conclude this part and point out possible future research directions in Section 5.8.

5.2 Network Model

We employ the network model defined in Part 3 as follows. We consider a probabilistic WSN consisting of n sensors, denoted by $s_1, s_2, \cdots, s_n$ respectively, and one sink deployed in a square area with size $A = cn$ (i.e., the node density of this WSN is $\frac{1}{c}$), where c is a constant. All the sensor nodes know their location information. Furthermore, we assume all the sensors are independent and identically distributed (i.i.d.) and without of generality, the

sink is located at the top-right corner of the square[2]. During each time interval, every sensor produces a data packet of B bits, and multiple data packets of the same snapshot can also be aggregated to a single data packet of B bits. All the transmissions are conducted over a common wireless channel with bandwidth W bits/second, i.e. the data transmission rate between any pair of nodes is at most W. We further assume the network time is synchronized and slotted into time slots of length $t_o = B/W$ seconds[3].

During the data transmission process, all the sensors work with a fixed power P. Therefore, when sensor s_i transmits a packet to sensor s_j, the Signal-to-Interference-plus-Noise Ratio (SINR) associated with s_i at s_j is defined as

$$\Lambda(s_i, s_j) = SINR(s_i, s_j) = \frac{P \cdot \|s_i - s_j\|^{-\alpha}}{N_0 + \sum_{k \neq i} P \cdot \|s_k - s_j\|^{-\alpha}}, \tag{5.1}$$

where, $\|s_i - s_j\|$ is the Euclidean distance between s_i and s_j, α is the path-loss exponent and usually $\alpha \in [3, 5]$, N_0 is a constant representing the background noise, and s_k is another concurrent sender other than s_i. To simplify the analysis, under the DNM, people usually assume that s_j can receive the data packet from s_i successfully if $\Lambda(s_i, s_j)$ is greater than a predefined value. However, in real applications, due to the existence of many lossy links, a successful data transmission between two nodes can be conducted with a probability instead of a fixed predetermined value. Therefore, a more practical and accurate method to depict WSNs is by a *Probabilistic Network Model* (PNM), where each link is associated with a *success probability* which indicates the probability that a successful data transmission can be conducted over this link. According to the empirical literatures [75], we define the success probability associated with s_i and s_j as

$$\Pr(s_i, s_j) = (1 - \eta_1 \cdot e^{-\eta_2 \cdot \Lambda(s_i, s_j)})^{\eta_3}, \tag{5.2}$$

[2]Note that it is easier to extend to the situation that the sink is located at anywhere else in the WSN, and we partition the WSN into four quadrants (taking the sink as the origin) and consider each quadrant individually.

[3]This assumption is reasonable since recent works, e.g. [81], showed that network-wide synchronization (at least at the millisecond level) is achievable.

where η_1, η_2, and $\eta_3 > 1$ are positive constants. Clearly, to successfully transmit a data packet to s_j, the expected number of transmission times of s_i satisfies a *geometric distribution* with parameter $\Pr(s_i, s_j)$, i.e. the expected number of time slots used to successfully transmit a data packet from s_i to s_j is $1/\Pr(s_i, s_j)$.

Actually, the successful probability of each link should not be too low, since a low successful probability implies many retransmission times and too much energy consumption until a successful transmission. Thus, similar as in Part 3, we define a *promising transmission threshold probability* p_o. Then, for any pair of nodes s_i and s_j, the data transmission between them can be initialized only if $\Pr(s_i, s_j) \geq p_o$. Now, for any qualified data transmission node pair, the expected number of transmission times to successfully transmit a data packet is at most $1/p_o$. Therefore, similar as in Part 3, we define a *normalized time slot* $t_n = t_o/p_o$ for convenience.

In the studied data aggregation problem, multiple data packets can be aggregated into one by applying a data aggregation function, e.g. MAX, MIN, SUM, etc. Formally, similar as in Part 4, we can define the SDA problem as follows. Let X and Y bet two subsets of $V = \{s_0, s_1, s_2, \cdots, s_n\}$, where s_0 is the sink node, and $X \cap Y = \emptyset$. The data of the nodes in X is said to be *aggregated* to the nodes in Y in a time slot if all the nodes in X can transmit their local aggregation data to the nodes in Y concurrently and interference-freely during that time slot. In this aggregation process, we call X a *transmitter set*. Then, the SDA problem can be defined as to seek a *SDA schedule* which consists of a sequence of transmitter sets $X_1, X_2, \cdots, X_M$, such that

1. $\forall 1 \leq i \neq j \leq M, X_i \cap X_j = \emptyset$;
2. $\bigcup_{i=1}^{M} X_i = V \setminus \{s_0\}$, where M is the latency of this SDA schedule;
3. Data can be aggregated from X_i to $V \setminus \bigcup_{j=1}^{i} X_j$ during time slot i for $i = 1, 2, \cdots, M$.

Based on the SDA problem, the definition of the CDA problem can be defined to seek an aggregation schedule for multiple continuous snapshots, with each snapshot corresponds to a schedule similar as in the SDA problem. Note that, the schedule for multiple snapshots in

the CDA problem may have some time overlap, i.e. the data aggregation in CDA may be pipelined.

According to the defined PNM, SDA problem, and CDA problem, we further formally define the data aggregation capacity as the ratio between the amount of data been aggregated and the time used to transmit the aggregated values of these data to the sink, i.e. SDA capacity is defined as nB/Γ, where Γ is the time used to transmit the aggregated value of a snapshot to the sink; to gather the aggregated value of each snapshot of N continuous snapshots (gathering N aggregated values to the sink, finally), the CDA capacity is defined as NnB/Γ, where now Γ is the time used to transmit the N aggregated values to the sink.

5.3 Network Partition

In this section, we partition a WSN into *cells* and *equivalent color classes* by the similar method used in Part 3.

5.3.1 Cell-Based Network Partition

Since we assume a WSN is deployed in a square area with $A = cn$, we partition this square into small square cells with side length $l = \sqrt{ce \log n}$ by horizontal and vertical lines starting at the left-bottom-most point. Moreover, we use $m = \sqrt{n/e \log n}$ to denote the number of cells in each row/column. For convenience, we also assign each cell a pair of coordinates (i, j) $(1 \leq i, j \leq m)$, where i and j indicate this cell is located at the i-th column and the j-th row respectively from the left-bottom-most point. Further, we use $\mathfrak{c}_{i,j}$ to denote the cell with coordinates (i, j). According to the communication mode of data aggregation and considering the fact that the sink is located at the top-right corner, we define four possible data transmission modes for the sensors in each cell, namely *inside transmission mode*, *upward transmission mode*, *rightward transmission mode*, and *up-rightward transmission mode*. Under the insider transmission mode, a node in $\mathfrak{c}_{i,j}$ transmits its data packet to another node also in $\mathfrak{c}_{i,j}$. Under the upward (rightward/up-rightward) transmission mode,

cell $\mathfrak{c}_{i,j}$ transmits its data packets to cell $\mathfrak{c}_{i,j+1}$ ($\mathfrak{c}_{i+1,j}/\mathfrak{c}_{i+1,j+1}$)[4].

For cell $\mathfrak{c}_{i,j}$, let the random variable $\mathcal{X}_{i,j}$ denote the number of sensors in it. Then, the expected number of sensors within $\mathfrak{c}_{i,j}$ $(1 \leq i, j \leq m)$, i.e. $\mathbb{E}[\mathcal{X}_{i,j}]$, can be determined by Lemma 5.3.1.

Lemma 5.3.1 $\mathbb{E}[\mathcal{X}_{i,j}] = e \log n$.

Proof: Since all the sensors are i.i.d., the number of sensors within a cell satisfies the *binomial distribution* with parameters $(n, \frac{l^2}{A})$. Thus, $\mathbb{E}[\mathcal{X}_{i,j}] = n \cdot \frac{l^2}{A} = e \log n$. □

Subsequently, we can obtain the upper and lower bounds of the number of sensors within cell $\mathfrak{c}_{i,j}$ $(1 \leq i, j \leq m)$ as shown in Lemma 5.3.2 and Lemma 5.3.3, respectively. The proof techniques of Lemma 5.3.2 and Lemma 5.3.3 are similar as that in the proof of Lemma 3.3.2 and Lemma 3.3.3.

Lemma 5.3.2 *For any cell* $\mathfrak{c}_{i,j}$ $(1 \leq i, j \leq m)$, $\Pr(\mathfrak{c}_{i,j}$ *contains* $6.7 \log n$ *sensors or more*$) = \Pr(\mathcal{X}_{i,j} \geq 6.7 \log n) \leq \frac{1}{n^2}$. *Then, it is almost sure that* $\mathfrak{c}_{i,j}$ *contains no more than* $6.7 \log n$ *sensors.*

Proof: Since $\mathcal{X}_{i,j}$ is a *binomial random variable* with parameters $(n, \frac{l^2}{A})$ as shown in Lemma 5.3.1, by applying the Chernoff bound and for any $\xi > 0$, we have

$$\Pr(\mathcal{X}_{i,j} \geq 6.7 \log n) \leq \min_{\xi>0} \frac{\mathbb{E}[\exp(\xi \mathcal{X}_{i,j})]}{\exp(6.7\xi \log n)} \tag{5.3}$$

$$= \min_{\xi>0} \frac{(1 + (e^{\xi} - 1) \cdot e \log n/n)^n}{\exp(6.7\xi \log n)} \tag{5.4}$$

$$\leq \min_{\xi>0} \frac{\exp((e^{\xi} - 1) \cdot e \log n)}{\exp(6.7\xi \log n)} \tag{5.5}$$

$$= \min_{\xi>0} \exp((e^{\xi+1} - e - 6.7\xi) \cdot \log n). \tag{5.6}$$

[4]For convenience, we use a cell and the sensors within this cell interchangeable in the following of this part.

Particularly, let $\xi = \ln 6.7 - 1$. We have

$$\Pr(\mathcal{X}_{i,j} \geq 6.7 \log n) \leq \exp(-2 \log n) \tag{5.7}$$

$$\leq \exp(-2 \ln n) \tag{5.8}$$

$$= \frac{1}{n^2}. \tag{5.9}$$

Since $\sum_{n>0} \frac{1}{n^2}$ is bounded by the result of the Basel problem, $\Pr(\chi_{i,j} \leq 6.7 \log n) \sim 1$ according to the Borel-Cantelli Lemma, i.e. it is almost sure that $\mathcal{X}_{i.j} \leq 6.7 \log n$. □

Lemma 5.3.3 *For any cell* $\mathfrak{c}_{i,j}$ $(1 \leq i, j \leq m)$, $\Pr(\mathfrak{c}_{i,j}$ *contains* $\frac{1}{2e} \log n$ *sensors or fewer*$) = \Pr(\mathcal{X}_{i,j} \leq \frac{1}{2e} \log n) \leq \frac{1}{n^2}$. *Then, it is almost sure that* $\mathfrak{c}_{i,j}$ *contains no fewer than* $\frac{1}{2e} \log n$ *sensors.*

Proof: Similar to the proof of Lemma 5.3.2, applying the Chernoff bound and for any $\xi < 0$, we have

$$\Pr(\mathcal{X}_{i,j} \leq \frac{1}{2e} \log n) \leq \min_{\xi<0} \exp((e^{\xi+1} - e - \frac{1}{2e}\xi) \cdot \log n). \tag{5.10}$$

Let $\xi = \ln \frac{1}{2e} - 1$, we have

$$\Pr(\mathcal{X}_{i,j} \leq \frac{1}{2e} \log n) \tag{5.11}$$

$$\leq \frac{1}{n^2}. \tag{5.12}$$

Thus, by the Borel-Cantelli Lemma, $\Pr(\chi_{i,j} \geq \frac{1}{2e} \log n) \sim 1$, i.e. it is almost sure that $\mathfrak{c}_{i,j}$ contains no fewer than $\frac{1}{2e} \log n$ sensors. □

From Lemma 5.3.1, we know that the average number of sensors within a cell is $e \log n$. From Lemma 5.3.2 and Lemma 5.3.3, we know that the probabilities that a cell contains more than $6.7 \log n$ sensors or fewer than $\frac{1}{2e} \log n$ sensors are zero for large n. Therefore, it is reasonable for us to use $6.7 \log n$ and $\frac{1}{2e} \log n$ as the upper and lower bounds as the number of sensors within a cell, respectively. In the following discussion, we assume a cell contains

$e \log n$ sensors in the *average case*, $6.7 \log n$ sensors in the *worst case*, and $\frac{1}{2e} \log n$ sensors in the *best case*.

5.3.2 Equivalent Color Class

After partitioning the WSN into cells, we further partition all the cells into disjoint *cell sets*, named *equivalent color classes*, by two vectors. For each equivalent color class, we assign it a *color* (actually assign this color to all the cells within this equivalent color class), denoted by a natural number. The two vectors we use to partition the cells are $\vec{X} = (\omega, 0)$ and $\vec{Y} = (0, \omega)$, where $\omega \in \mathbb{N}^+$ is a constant positive integer. Based on $\vec{X}$ and $\vec{Y}$, we define the *equivalent color class* containing $\mathfrak{c}_{i,j}$ as $\{\mathfrak{c}_{x,y} | (x, y) = (i, j) + a \cdot \vec{X} + b \cdot \vec{Y}, x \in [1, m], y \in [1, m], a \in \mathbb{Z}, b \in \mathbb{Z}\}$. Within an equivalent color class, if cell $\mathfrak{c}_{i,j}$ has the smallest distance to the left-bottom-most point, $\mathfrak{c}_{i,j}$ is called the *pivot cell* of this class. Further, the equivalent color class having $\mathfrak{c}_{i,j}$ as the pivot cell is denoted by $\mathbb{C}_{i,j}$.

Based on the equivalent color class partition method, it is straightforward to obtain the following lemma.

Lemma 5.3.4 *The cells of a WSN can be partitioned into ω^2 equivalent color classes.*

Let $\Lambda(\mathbb{C}_{i,j}) = \min\{\Lambda(s_u, s_v) | s_u$ is any sensor in any cell of $\mathbb{C}_{i,j}$, s_v is the destination node of s_u under any transmission mode$\}$. Then, we have the following lemma. Lemma 5.3.5 can be proven by similar techniques in Lemma 3.3.4.

Lemma 5.3.5 *Let $R = \omega l$. If all the cells not in $\mathbb{C}_{i,j}$ keep silent and all the cells within $\mathbb{C}_{i,j}$ conduct data transmissions concurrently and successfully*[5]*, then $\Lambda(\mathbb{C}_{i,j}) \geq \frac{P \cdot d^{-\alpha}}{N_0 + P \cdot \varpi \cdot R^{-\alpha}}$, where $d \leq 2\sqrt{2}l$ is the distance between a communication pair and $\varpi \approx 8 \cdot (3^\alpha + 2.847)$ is a positive constant.*

Based on Lemma 5.3.5, we can determine the value of ω to make all the cells within each equivalent color class conduct transmissions concurrently and successfully as shown in

[5] *Here, "successfully" means all the data transmissions conducted are promising transmissions under any transmission mode.*

Theorem 5.3.1. Theorem 5.3.1 can be proven by similar techniques in Lemma 3.3.5 and Theorem 3.3.1.

Theorem 5.3.1 *If we properly set* $\omega = \Theta(\frac{d+\Theta(1)}{l})$ *and all the other cells not in* $\mathbb{C}_{i,j}$ *keep silent, then all the cells in* $\mathbb{C}_{i,j}$ $(1 \leq i, j \leq \omega)$ *can conduct data transmissions under any communication mode concurrently and successfully.*

Based on Theorem 5.3.1, we assign an appropriate value for ω, i.e. $\Theta(\frac{d+\Theta(1)}{l})$, in the following discussion, which implies that all the cells in equivalent color class $\mathbb{C}_{i,j}$ $(1 \leq i, j \leq m)$ can conduct data transmissions under any communication mode concurrently and successfully.

5.4 Snapshot Data Aggregation

In this section, we consider the snapshot data aggregation problem, propose a Cell-based Aggregation Scheduling (CAS) algorithm for snapshot data aggregation, and analyze the achievable network capacity of CAS. Furthermore, we also derive the upper bound network capacity of the snapshot data aggregation problem, which shows our proposed CAS is order-optimal.

5.4.1 Cell-Based Snapshot Data Aggregation

As proven in Section 5.3.1, each cell $\mathfrak{c}_{i,j}$ $(1 \leq i, j \leq m)$ contains $e \log n$, $6.7 \log n$, and $\frac{1}{2e} \log n$ sensors in the average case, the worst case, and the best case, respectively. Therefore, we define a *super time slot*, denoted by t_s, for convenience, where $t_s = e \log n \cdot t_n$, $t_s = 6.7 \log n \cdot t_n$, and $t_s = \frac{1}{2e} \log n \cdot t_n$ in the average case, the worst case, and the best case, respectively. Thus, within a super time slot, all the sensors within a cell can be assigned one normalized time slot to transmit its data. Then, we design a two-phase snapshot data aggregation algorithm, named *Cell-based Aggregation Scheduling* (CAS), as follows.

Intra-Cell Scheduling Phase. In this phase, we schedule the data aggregation operations within each cell. First, for cell $\mathfrak{c}_{i,j}$ $(1 \leq i, j \leq m)$, we choose one sensor from this

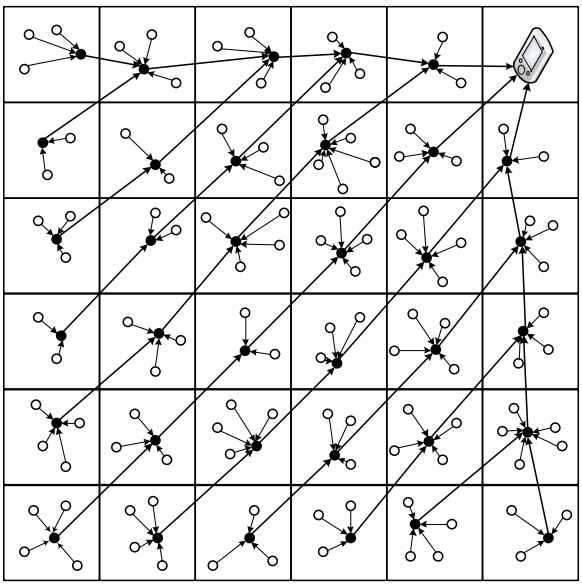


Figure 5.1 Data aggregation tree.

cell as the *local aggregation node* of this cell, denoted by $\mathcal{A}_{i,j}$. As shown in Figure 5.1, the black node within each cell is the local aggregation node of that cell. Subsequently, within each cell $\mathfrak{c}_{i,j}$ $(1 \leq i, j \leq m)$, all the non-local aggregation nodes transmit their data to $\mathcal{A}_{i,j}$ under the inside transmission mode according to a sequential order, e.g. sensors with smaller ID transmit first. Finally, $\mathcal{A}_{i,j}$ $(1 \leq i, j \leq m)$ aggregates all the data it received and its own data to form a new aggregated data packet to transmit in the second phase. In Section 5.3.2, we have partitioned all the cells into ω^2 equivalent color classes and assigned each $\mathbb{C}_{x,y}$ $(1 \leq x, y \leq \omega)$ a color $x + (y-1)\omega$. Moreover, all the cells within an equivalent color class can conduct data transmissions under any transmission mode concurrently and successfully according to Theorem 5.3.1. Thus, to schedule all the cells to finish the intra-cell scheduling phase, we can schedule each equivalent color class for one super time slot, with $\mathbb{C}_{x,y}$ scheduled in the $(x + (y-1)\omega)$-th super time slot.

After the first phase, all the local aggregation nodes need to transmit the local aggregated values to the sink to obtain the final aggregation value of the whole snapshot (note that data can also be aggregated during the transmission process). To finish the data aggregation task in the second phase, we construct a data aggregation tree, denoted by $\mathcal{T}$, rooted at the

sink to connect all the local aggregation nodes according to the similar rules as in Part 3 to construct a data collection tree:

- For $\mathcal{A}_{i,j}$ $(1 \leq i, j \leq m-1)$, it transmits its data to $\mathcal{A}_{i+1,j+1}$ under the up-rightward transmission mode, i.e. connect $\mathcal{A}_{i,j}$ with $\mathcal{A}_{i+1,j+1}$ as its parent node;
- For $\mathcal{A}_{m,j}$ $(1 \leq j \leq m-1)$, it transmits its data to $\mathcal{A}_{m,j+1}$ under the upward transmission mode, i.e. connect $\mathcal{A}_{m,j}$ with $\mathcal{A}_{m,j+1}$ as its parent node;
- For $\mathcal{A}_{i,m}$ $(1 \leq i \leq m-1)$, it transmits its data to $\mathcal{A}_{i+1,m}$ under the rightward transmission mode, i.e. connect $\mathcal{A}_{i,m}$ with $\mathcal{A}_{i+1,m}$ as its parent node.

An example data aggregation tree is shown in Figure 5.1. For $\mathcal{A}_{i,j}$, we define the *level* of $\mathcal{A}_{i,j}$ in $\mathcal{T}$, denoted by $h_{i,j}$, as the number of hops from $\mathcal{A}_{i,j}$ to the root of $\mathcal{T}$ (i.e. the sink). Clearly, $\mathcal{T}$ has $m-1$ levels. Furthermore, we denote the set of all the local aggregation nodes with the same level k $(1 \leq k \leq m-1)$ as $\mathcal{L}_k$, i.e. $\mathcal{L}_k = \{\mathcal{A}_{i,j} | h_{i,j} = k\}$. For the local aggregation nodes in $\mathcal{L}_k$ $(1 \leq k \leq m-1)$, suppose they come from $\mathcal{C}_k$ equivalent color classes. Then, we gather the final aggregation value of a snapshot as shown in the second phase.

Inter-Cell Scheduling Phase. In this phase, we schedule local aggregation nodes level by level, staring from the $(m-1)$-th level. For every local aggregation node in $\mathcal{L}_k$, after it receives the aggregation values from its children local aggregation nodes in $\mathcal{L}_{k+1}$, it aggregates the received values with its own data to form a new data packet. Subsequently, it transmits the new obtained data packet to its parent local aggregation node in $\mathcal{L}_{k-1}$ during its available time slots. Since the local aggregation nodes in $\mathcal{L}_k$ $(1 \leq k \leq m-1)$ come from $\mathcal{C}_k$ equivalent color classes, they can be scheduled by $\mathcal{C}_k$ normalized time slots.

At the end of the second phase, the sink will receive partial aggregated values of a snapshot. Consequently, the sink can obtain the final aggregation value of a snapshot by doing some aggregation calculations.

5.4.2 Capacity Analysis of CAS

In this subsection, we analyze the achievable network capacity of CAS in the worst case, the average case and the best case, respectively. Subsequently, we study the upper bound capacity of the snapshot data aggregation problem, which implies the achievable capacities of CAS in all the cases are order-optimal.

Lemma 5.4.1 *For snapshot data aggregation, the number of normalized time slots used by CAS is at most* $6.7\omega^2 \log n + (2\omega - 1)(m - 1)$ *in the worst case,* $e\omega^2 \log n + (2\omega - 1)(m - 1)$ *in the average case, and* $\frac{1}{2e}\omega^2 \log n + (2\omega - 1)(m - 1)$ *in the best case, respectively.*

Proof: First, it is straightforward that in the first phase of CAS, the number of super time slots used is ω^2. According to the definition of a super time slot t_u, it follows that the number of normalized time slots used by CAS in the first phase is $6.7\omega^2 \log n$ in the worst case, $e\omega^2 \log n$ in the average case, and $\frac{1}{2e}\omega^2 \log n$ in the best case.

In the second phase of CAS, the local aggregation nodes in $\mathcal{T}$ are scheduled level by level, and level $\mathcal{L}_k$ will cost $\mathcal{C}_k$ normalized time slots. In $\mathcal{L}_k = \{\mathcal{A}_{m-k,j} | m - k \leq j \leq k\} \cup \{\mathcal{A}_{j,m-k} | m - k \leq j \leq k\}$, both $\{\mathcal{A}_{m-k,j} | m - k \leq j \leq k\}$ and $\{\mathcal{A}_{j,m-k} | m - k \leq j \leq k\}$ come from at most ω equivalent color classes. $\mathcal{A}_{m-k,m-k}$ can be from only one equivalent color class, which implies $\mathcal{C}_k \leq 2\omega - 1$. Furthermore, $\mathcal{T}$ has $m - 1$ levels, which implies the second phase of CAS can be done with at most $(2\omega - 1)(m - 1)$ normalized time slots. In summary, Lemma 5.4.1 holds. □

Based on Lemma 5.4.1, we can obtain the achievable network capacities of CAS in different cases as shown in Theorem 5.4.1.

Theorem 5.4.1 *For snapshot data aggregation, the achievable network capacity of CAS is*

$$\Omega(\frac{p_o n}{6.7\omega^2 \log n + 2\omega\sqrt{n/e \log n}} \cdot W) = \Omega(\frac{p_o\sqrt{en \log n}}{2\omega} \cdot W)$$

in the worst case,

$$\Omega(\frac{p_o n}{e\omega^2 \log n + 2\omega\sqrt{n/e \log n}} \cdot W) = \Omega(\frac{p_o\sqrt{en \log n}}{2\omega} \cdot W)$$

in the average case, and

$$\Omega(\frac{p_o n}{\frac{1}{2e}\omega^2 \log n + 2\omega\sqrt{n/e \log n}} \cdot W) = \Omega(\frac{p_o\sqrt{en \log n}}{2\omega} \cdot W)$$

in the average case.

Proof: In the worst case, the achievable network capacity of CSA is

$$\frac{nB}{(6.7\omega^2 \log n + (2\omega - 1)(m-1)) \cdot t_n} \tag{5.13}$$

$$\geq \frac{p_o nB}{(6.7\omega^2 \log n + 2\omega m)t_o} \tag{5.14}$$

$$= \Omega(\frac{p_o n}{6.7\omega^2 \log n + 2\omega\sqrt{n/e \log n}} \cdot W) \tag{5.15}$$

$$= \Omega(\frac{p_o\sqrt{en \log n}}{2\omega} \cdot W). \tag{5.16}$$

Similarly, the achievable capacities of CAS in the average case and best case can be obtained. □

Now, we study the upper bound capacity of the snapshot data aggregation problem as shown in Theorem 5.4.2, which is an inherent property of snapshot data aggregation.

Theorem 5.4.2 *The upper bound capacity of the snapshot data aggregation problem is at most* $O(\frac{p_o\sqrt{en \log n}}{3} \cdot W)$*, which implies that CAS has successfully achieved order optimal capacities in every case.*

Proof: First, to aggregate the data produced at cells $\mathfrak{c}_{i,1}$ $(1 \leq i \leq m)$ and $\mathfrak{c}_{1,j}$ $(1 \leq j \leq m)$, we need at least $\frac{1}{2e}\log n$ normalized time slots no matter what scheduling algorithm we use. Second, since it is easy to know that for any $\mathcal{A}_{i,j} \in \mathcal{L}_k$ ($i = k$ or $j = k$) whose parent node is $\mathcal{A}_{i+1,j+1}$, both $\Lambda(\mathcal{A}_{i+1,j}, \mathcal{A}_{i+1,j+1})$ and $\Lambda(\mathcal{A}_{i+2,j}, \mathcal{A}_{i+1,j+1})$ are no less than $\Lambda(\mathcal{A}_{i,j}, \mathcal{A}_{i+1,j+1})$

at $\mathcal{A}_{i+1,j+1}$, as well as $\Lambda(\mathcal{A}_{i,j+1}, \mathcal{A}_{i+1,j+1})$ and $\Lambda(\mathcal{A}_{i,j+2}, \mathcal{A}_{i+1,j+1})$. It follows that $\mathcal{C}_k \geq 3$ for every $1 \leq k \leq m-1$. This further implies that we need at least $3(m-1)$ normalized time slots to transmit the aggregated values at $\mathcal{A}_{i,1}$ $(1 \leq i \leq m)$ and $\mathcal{A}_{1,j}$ $(1 \leq j \leq m)$ to the sink. In summary, the number of normalized time slots used to obtain the final aggregation value of a snapshot is at least $\frac{1}{2e}\log n + 3(m-1)$, which implies the upper bound capacity of the snapshot data aggregation problem is at most

$$\frac{nB}{(\frac{1}{2e}\log n + 3(m-1)) \cdot t_n} \tag{5.17}$$

$$= \frac{p_o n}{\frac{1}{2e}\log n + 3(\sqrt{n/e \log n} - 1)} \cdot W \tag{5.18}$$

$$= O(\frac{p_o\sqrt{en\log n}}{3} \cdot W). \tag{5.19}$$

Since the achievable capacities of CAS in every case are $O(\frac{p_o\sqrt{en\log n}}{2\omega} \cdot W)$, CAS has successfully achieved order optimal capacities. □

5.5 Continuous Data Aggregation

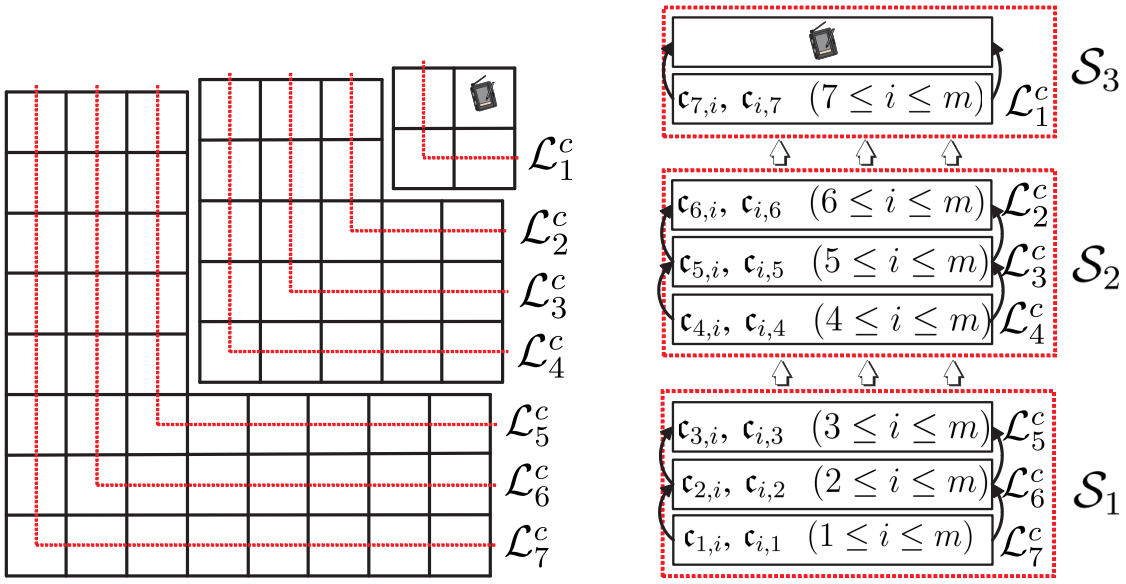
To address the continuous data aggregation problem, we design a Level-based Aggregation Scheduling (LAS) algorithm in this section. Firstly, LAS partitions the data aggregation tree $\mathcal{T}$ (constructed in Section 5.4.1) into *segments* (as the segments in Part 3) and *cell-level classes*. Subsequently, LAS forms a data aggregation pipeline on the segments by scheduling the data aggregations of a level class concurrently. Furthermore, we also analyze the achievable capacities of LAS in every case, as well as the upper bound capacity of the continuous data aggregation problem, which implies that LAS has successfully achieved order-optimal capacities.

5.5.1 Level-based Aggregation Scheduling

In Section 5.4.1, we partition the local aggregation nodes $\mathcal{A}_{i,j}$ $(1 \leq i,j \leq m)$ on $\mathcal{T}$ into k levels, denoted by $\mathcal{L}_k$ $(1 \leq k \leq m-1)$. Since $\mathcal{A}_{i,j}$ corresponds to cell $\mathfrak{c}_{i,j}$ $(\mathcal{A}_{i,j}$

is the local aggregation node of $\mathfrak{c}_{i,j}$), we also define a *cell-level* $\mathcal{L}_k^c$ as $\mathcal{L}_k^c = \{\mathfrak{c}_{i,j} | h_{i,j} = k\}$ $(1 \leq k \leq m-1)$. Then, to form a data aggregation pipeline, we partition the $m-1$ cell-levels into $\lceil \frac{m-1}{\omega} \rceil$ *segments* with segment $\mathcal{S}_\iota = \{\mathcal{L}_k^c | m - (\iota - 1) \cdot \omega - 1 \geq k \geq \max\{1, m - \iota \cdot \omega\}\}$ $(1 \leq \iota \leq \lceil \frac{m-1}{\omega} \rceil)$. For instance, the data aggregation tree corresponding to the WSN shown in Figure 3.2 has 7 cell-levels, e.g. $\mathcal{L}_4^c = \{\mathfrak{c}_{4,4}, \mathfrak{c}_{4,5}, \mathfrak{c}_{4,6}, \mathfrak{c}_{4,7}, \mathfrak{c}_{4,8}, \mathfrak{c}_{5,4}, \mathfrak{c}_{6,4}, \mathfrak{c}_{7,4}, \mathfrak{c}_{8,4}\}$ and $\mathcal{L}_2^c = \{\mathfrak{c}_{6,6}, \mathfrak{c}_{6,7}, \mathfrak{c}_{6,8}, \mathfrak{c}_{7,6}, \mathfrak{c}_{8,6}\}$ as shown in Figure 5.2. If $\omega = 3$, these cell-levels can be partitioned into three segments with $\mathcal{S}_1 = \{\mathcal{L}_7^c, \mathcal{L}_6^c, \mathcal{L}_5^c\}$, $\mathcal{S}_2 = \{\mathcal{L}_4^c, \mathcal{L}_3^c, \mathcal{L}_2^c\}$, and $\mathcal{S}_3 = \{\mathcal{L}_1^c\}$ as shown in Figure 5.2(a), respectively. Furthermore, we also partition cell-levels into ω *cell-level classes*, with each cell-level class defined by $\mathbb{L}_g = \{\mathcal{L}_k^c | k\%\omega = g\}$ $(1 \leq g \leq \omega)$. For instance, the network shown in Figure 5.2(a) has three cell-level classes: $\mathbb{L}_1 = \{\mathcal{L}_7^c, \mathcal{L}_4^c, \mathcal{L}_1^c\}$, $\mathbb{L}_2 = \{\mathcal{L}_5^c, \mathcal{L}_2^c\}$, and $\mathbb{L}_0 = \{\mathcal{L}_6^c, \mathcal{L}_3^c\}$. Based on the definitions of segments and cell-level classes, it is clear that (i) each segment has ω cell-levels (only $\mathcal{S}_{\lceil (m-1)/\omega \rceil}$ may have fewer cell-levels); (ii) all the cell-levels within any segment belong to different cell-level classes, i.e. each segment contains exactly one cell-level from each of the ω cell-level classes; (iii) according to the definitions of equivalent color classes and cell-level classes, the cells within a cell-level class come from at most $2\omega - 1$ equivalent color classes. Now, to collect the aggregation value of each of N continuous snapshots, we are ready to propose our *Level-based Aggregation Scheduling* (LAS) algorithm. We explain the idea of LAS in a hierarchical way, from a coarse granularity to a subtle granularity, as follows.

Segment-Granularity Scheduling. Since a WSN has been partitioned into segments, a data aggregation/tranmission pipeline on these segments can be formed if we take each segment as a unit. Suppose for segment $\mathcal{S}_\iota$ $(1 \leq \iota \leq \lceil \frac{m-1}{\omega} \rceil)$, the number of normalized time slots used to transmit the aggregation values of a snapshot to the subsequent segment (to the sink for $\mathcal{S}_{\lceil (m-1)/\omega \rceil}$) is t_ι^n and define $t_p = \max\{t_\iota^n | 1 \leq \iota \leq \lceil \frac{m-1}{\omega} \rceil\}$. Then, to gather the N aggregation values of N continuous snapshots, a data aggregation/tansmission pipeline can be formed on these $\lceil \frac{m-1}{\omega} \rceil$ segments, with each segment working for t_p normalized time slots to transmit the aggregation values for each snapshot. Particularly, for segment $\mathcal{S}_\iota$, after it transmits the aggregation values of the j-th snapshot to segment $\mathcal{S}_{\iota+1}$ in t_p normalized time



(a) Segment partition. In this example, we assume $\omega = 3$, which implies this network can be partitioned into 3 segments: $\mathcal{S}_1 = \{\mathcal{L}_7^c, \mathcal{L}_6^c, \mathcal{L}_5^c\}$, $\mathcal{S}_2 = \{\mathcal{L}_4^c, \mathcal{L}_3^c, \mathcal{L}_2^c\}$, and $\mathcal{S}_3 = \{\mathcal{L}_1^c\}$

(b) Segment-granularity and cell-granularity scheduling.

Figure 5.2 Level-based aggregation scheduling.

slots (which also implies that $\mathcal{S}_\iota$ has already received the aggregation values of the $(j+1)$-th snapshot from segment $\mathcal{S}_{\iota-1}$), it starts to aggregate and transmit values for the $(j+1)$-th snapshot. For instance, the data aggregation pipeline formed on the three segments in Figure 5.2(a) is shown in Figure 5.2(b).

Level-Granularity Scheduling. Within each segment, LAS schedules data aggregation and transmission cell-level class by cell-level class, i.e. level by level. Taking the data aggregation/transmission process of the j-th snapshot in segment $\mathcal{S}_1 = \{\mathcal{L}_{m-1}^c, \mathcal{L}_{m-2}^c, \cdots, \mathcal{L}_{m-\omega}^c\}$ as an example, LAS first schedules $\mathcal{L}_{m-1}^c$ to transmit the aggregation values of the j-th snapshot to $\mathcal{L}_{m-2}^c$. Subsequently, after $\mathcal{L}_{m-2}^c$ receives the aggregation values of the j-th snapshot from $\mathcal{L}_{m-1}^c$, it aggregates the received values with its own data and transmits the new obtained aggregation values to the subsequent cell-level. This process is repeated until $\mathcal{L}_{m-\omega}^c$ transmits the aggregation values of the j-th snapshot to next segment. Since every segment does the same scheduling and the cell-levels have been partitioned into cell-level classes, the level-granularity scheduling is equivalent to schedule cell-level classes

repeatedly according to the order $\mathbb{L}_{(m-1)\%\omega}, \mathbb{L}_{(m-2)\%\omega}, \cdots, \mathbb{L}_{(m-\omega)\%\omega}$. Furthermore, since the cells within any cell-level class come from at most $2\omega - 1$ equivalent color classes as mentioned before, a cell-level class can be scheduled within $2\omega - 1$ super time slots. For the data aggregation pipeline shown in Figure 5.2(b), the cells in $\mathbb{L}_1 = \{\mathcal{L}_7^c, \mathcal{L}_4^c, \mathcal{L}_1^c\}$ will be scheduled simultaneously to transmit data equivalent color class by equivalent color class to the cells in $\mathbb{L}_0 = \{\mathcal{L}_6^c, \mathcal{L}_3^c\}$. Similarly, the data flow will be transmitted from cells in $\mathbb{L}_0 = \{\mathcal{L}_6^c, \mathcal{L}_3^c\}$ to cells in $\mathbb{L}_2 = \{\mathcal{L}_5^c, \mathcal{L}_2^c\}$, and then from $\mathbb{L}_2$ to $\mathbb{L}_1$.

Cell-Granularity Scheduling. Within each cell-level $\mathcal{L}_k^c$ $(1 \leq k \leq m-1)$, we have $2k+1$ cells which come from at most $2\omega - 1$ equivalent color classes as explained in Lemma 5.4.1. This further implies all the cells in $\mathcal{L}_k^c$ can be scheduled in $2\omega - 1$ super time slots. For cell $\mathfrak{c}_{i,j}$, during its available super time slot, it does similar operations as in CAS, i.e. all the non-local aggregation nodes transmit their data to $\mathcal{A}_{i,j}$, and then $\mathcal{A}_{i,j}$ transmits the aggregation value of cell $\mathfrak{c}_{i,j}$ to its parent node in $\mathcal{T}$. For instance, in cell-level $\mathcal{L}_7^c$ shown in Figure 5.2(b), all the cells come from 5 equivalent color classes: $\{\mathfrak{c}_{1,1}, \mathfrak{c}_{4,1}, \mathfrak{c}_{7,1}, \mathfrak{c}_{1,4}, \mathfrak{c}_{1,7}\}$, $\{\mathfrak{c}_{2,1}, \mathfrak{c}_{5,1}, \mathfrak{c}_{8,1}\}$, $\{\mathfrak{c}_{3,1}, \mathfrak{c}_{6,1}\}$, $\{\mathfrak{c}_{1,2}, \mathfrak{c}_{1,5}, \mathfrak{c}_{1,8}\}$, and $\{\mathfrak{c}_{1,3}, \mathfrak{c}_{1,6}\}$. These equivalent color classes of each cell-level will be scheduled one by one. For all the cells in each equivalent color class, they will be scheduled simultaneously as in CAS.

5.5.2 Capacity Analysis of LAS

Lemma 5.5.1 *In LAS, $t_p \leq 6.7\omega(2\omega - 1)\log n$ in the worst case; $t_p \leq e\omega(2\omega - 1)\log n$ in the average case; $t_p \leq \frac{1}{2e}\omega(2\omega - 1)\log n$ in the best case.*

Proof: As proven in Lemma 5.4.1, the cells within each cell-level come from at most $2\omega - 1$ equivalent color classes, which implies a cell-level can be scheduled in $2\omega - 1$ super time slots, i.e. $6.7(2\omega - 1)\log n$ normalized time slots in the worst case. Furthermore, each segment contains at most ω cell-levels, which implies that $t_p \leq 6.7\omega(2\omega - 1)\log n$ in the worst case. By similar reasons, Lemma 5.5.1 also holds in the average case and the best case. □

Based on Lemma 5.5.1, we can obtain the achievable network capacities of LAS in every

case as shown in Theorem 5.5.1.

Theorem 5.5.1 *To gather the aggregation value of each of N continuous snapshots, the achievable capacity of LAS is*

$$\begin{cases} \Omega(\frac{\sqrt{e}p_oN}{13.4\omega}\sqrt{\frac{n}{\log n}}\cdot W), & if\ N = O(\sqrt{\frac{n}{\log n}}) \\ \Omega(\frac{p_o}{13.4\omega^2}\frac{n}{\log n}\cdot W), & if\ N = \Omega(\sqrt{\frac{n}{\log n}}) \end{cases}$$

in the worst case,

$$\begin{cases} \Omega(\frac{p_oN}{2\sqrt{e}\omega}\sqrt{\frac{n}{\log n}}\cdot W), & if\ N = O(\sqrt{\frac{n}{\log n}}) \\ \Omega(\frac{p_o}{2e\omega^2}\frac{n}{\log n}\cdot W), & if\ N = \Omega(\sqrt{\frac{n}{\log n}}) \end{cases}$$

in the average case, and

$$\begin{cases} \Omega(\frac{e\sqrt{e}p_oN}{\omega}\sqrt{\frac{n}{\log n}}\cdot W), & if\ N = O(\sqrt{\frac{n}{\log n}}) \\ \Omega(\frac{ep_o}{\omega^2}\frac{n}{\log n}\cdot W), & if\ N = \Omega(\sqrt{\frac{n}{\log n}}) \end{cases}$$

in the best case.

Proof: From Lemma 5.5.1, $t_p \leq 6.7\omega(2\omega-1)\log n$ in the worst case, which implies that it takes at most $\lceil\frac{m}{\omega}\rceil\cdot 6.7\omega(2\omega-1)\log n$ normalized time slots to gather the aggregation values of the first snapshot by the sink. After that, according to the pipeline scheduling of LAS, the sink will receive the aggregation values of a subsequent snapshot every $6.7\omega(2\omega-1)\log n$ normalized time slots, until the aggregation values of all the N continuous snapshots have been gathered by the sink. Therefore, LAS uses at most

$$\lceil\frac{m}{\omega}\rceil\cdot 6.7\omega(2\omega-1)\log n + (N-1)\cdot 6.7\omega(2\omega-1)\log n \tag{5.20}$$

$$= O(13.4\omega\sqrt{n\log n/e} + 13.4\omega^2 N\log n) \tag{5.21}$$

normalized time slots to gather all the aggregation values of N continuous snapshots. It

follows that the achievable capacity of LAS in the worst case is

$$\frac{nNB}{O(13.4\omega\sqrt{n\log n/e}+13.4\omega^2 N\log n)\cdot t_n} \tag{5.22}$$

$$= \frac{p_o nN}{O(13.4\omega\sqrt{n\log n/e}+13.4\omega^2 N\log n)}\cdot W \tag{5.23}$$

$$= \begin{cases} \Omega(\frac{\sqrt{e}p_o N}{13.4\omega}\sqrt{\frac{n}{\log n}}\cdot W), & if\ N = O(\sqrt{\frac{n}{\log n}}) \\ \Omega(\frac{p_o}{13.4\omega^2}\frac{n}{\log n}\cdot W), & if\ N = \Omega(\sqrt{\frac{n}{\log n}}) \end{cases}. \tag{5.24}$$

By a similar method, we can obtain the achievable capacities of LAS in the average case and in the best case. □

Now, we study the upper bound capacity of the continuous data aggregation problem as shown in Theorem 5.5.2, which implies that LAS has already successfully achieved order optimal capacities in every case.

Theorem 5.5.2 *The upper bound capacity of the continuous data aggregation problem to collect the aggregation values of N continuous snapshots is*

$$\begin{cases} O(\frac{2e\sqrt{e}p_o N}{3}\sqrt{\frac{n}{\log n}}\cdot W), & if\ N = O(\sqrt{\frac{n}{\log n}}) \\ O(\frac{2ep_o}{9}\frac{n}{\log n}\cdot W), & if\ N = \Omega(\sqrt{\frac{n}{\log n}}) \end{cases}.$$

Proof: As proven in Theorem 5.4.2, the local aggregation nodes (cells) of each level (cell-level) come from at least 3 equivalent color classes, which implies that it takes at least 3 super time slots to schedule a cell-level. Furthermore, because of the same reason, the cell-levels can be partitioned into segments with each segment contains at least 3 levels. Therefore, $t_p \geq \frac{9}{2e}\log n$. Then, the number of normalized time slots used to gather the aggregation values of N continuous snapshots is at least

$$\left\lceil\frac{m}{3}\right\rceil\cdot\frac{9}{2e}\log n + (N-1)\cdot\frac{9}{2e}\log n \tag{5.25}$$

$$= \Omega(\frac{3}{2e}\sqrt{\frac{n\log n}{e}}+\frac{9N}{2e}\log n). \tag{5.26}$$

Thus, the upper bound capacity of the continuous data aggregation problem is

$$\frac{nNB}{\Omega(\frac{3}{2e}\sqrt{\frac{n\log n}{e}}+\frac{9N}{2e}\log n)\cdot t_n} \tag{5.27}$$

$$=\frac{p_o nN}{\Omega(\frac{3}{2e}\sqrt{\frac{n\log n}{e}}+\frac{9N}{2e}\log n)}\cdot W \tag{5.28}$$

$$=\begin{cases} O(\frac{2e\sqrt{e}p_o N}{3}\sqrt{\frac{n}{\log n}}\cdot W), & if\ N=O(\sqrt{\frac{n}{\log n}}) \\ O(\frac{2ep_o}{9}\frac{n}{\log n}\cdot W), & if\ N=\Omega(\sqrt{\frac{n}{\log n}}) \end{cases}, \tag{5.29}$$

which implies that the achievable capacities of LAS in every case are order optimal according to Theorem 5.5.1. □

5.6 Discussion: Capacity of CAS and LAS under Non-I.I.D. Models

Assuming the network is distributed according to an i.i.d. model is convenient for algorithm design and analyzing the achievable data aggregation capacity of proposed algorithms. However, this assumption may not hold in some situations. Therefore, in this section, we analyze the capacity performance of CAS and LAS under non-i.i.d. models. Specifically, we consider that all the sensor nodes are deployed according to a *stationary Poisson point process* in this section.

Similar as in Section 5.2, we assume n sensor nodes deployed in a square area of size $A = cn$ according to a stationary Poisson point process with parameter λ_p. Subsequently, by the same network partition method in Section 5.3.1, we partition the network into cells with side length $l = \sqrt{ce\log n}$. Then, for cell $\mathfrak{c}_{i,j}$, the expected number of sensor nodes in

$\mathfrak{c}_{i,j}$ is

$$\mathbb{E}[\mathcal{X}_{i,j}] = \sum_{k=1}^{+\infty} \Pr(\mathcal{X}_{i,j} = k) \cdot k \tag{5.30}$$

$$= \sum_{k=1}^{+\infty} \frac{(\lambda_p l^2)^k}{k!} \exp(-\lambda_p l^2) \cdot k \tag{5.31}$$

$$= \lambda_p l^2 \tag{5.32}$$

$$= ce\lambda_p \log n. \tag{5.33}$$

According to $\mathbb{E}[\mathcal{X}_{i,j}]$, we can prove the following conclusions by similar techniques in Lemma 5.3.2 and Lemma 5.3.3.

Lemma 5.6.1 *For any cell $\mathfrak{c}_{i,j}$ and a constant value $a = \arg\min_{\xi>0} \frac{ce^{\xi+1}\lambda_p - ce\lambda_p + 2}{\xi}$, $\Pr(\mathfrak{c}_{i,j}$ contains $a \log n$ sensors or more$) = \Pr(\mathcal{X}_{i,j} \geq a \log n) \leq \frac{1}{n^2}$, which implies it is almost sure that $\mathfrak{c}_{i,j}$ contains no more than $a \log n$ sensor nodes.*

Proof Sketch: Based on $\mathbb{E}[\mathcal{X}_{i,j}]$ and applying the Chernoff bound and for any $\xi > 0$, we have

$$\Pr(\mathcal{X}_{i,j} \geq a \log n) = \Pr(e^{\xi \mathcal{X}_{i,j}} \geq e^{\xi \cdot a \log n}) \tag{5.34}$$

$$\leq \min_{\xi>0} \frac{\mathbb{E}[e^{\xi \mathcal{X}_{i,j}}]}{e^{\xi \cdot a \log n}} \tag{5.35}$$

$$= \min_{\xi>0} \frac{e^{ce\lambda_p \log n \cdot (e^{\xi}-1)}}{e^{\xi \cdot a \log n}} \tag{5.36}$$

$$= \min_{\xi>0} e^{(ce^{\xi+1}\lambda_p - ce\lambda_p - \xi a)\log n} \tag{5.37}$$

$$= e^{-2 \log n} \tag{5.38}$$

$$\leq e^{-2 \ln n} \tag{5.39}$$

$$= \frac{1}{n^2}. \tag{5.40}$$

Since $\sum_{n>0} \frac{1}{n^2}$ is bounded, $\Pr(\chi_{i,j} \leq 6.7 \log n) \sim 1$ according to the Borel-Cantelli Lemma, i.e. it is almost sure that $\mathcal{X}_{i.j} \leq 6.7 \log n$. □

From Lemma 5.6.1, we know that the number of sensor nodes within a cell is upper bounded by $a\log n$ with probability 1, where $a = \arg\min_{\xi>0} \frac{ce^{\xi+1}\lambda_p - ce\lambda_p + 2}{\xi}$. Similarly, we can also derive the lower bound of the number of sensor nodes within a cell as follows.

Lemma 5.6.2 *For any cell $\mathfrak{c}_{i,j}$ and a constant value $b = \arg\max_{\xi<0} \frac{ce^{\xi+1}\lambda_p - ce\lambda_p + 2}{\xi}$, $\Pr(\mathfrak{c}_{i,j}$ contains $b\log n$ sensors or less$) = \Pr(\mathcal{X}_{i,j} \leq b\log n) \leq \frac{1}{n^2}$, which implies it is almost sure that $\mathfrak{c}_{i,j}$ contains no less than $b\log n$ sensor nodes.*

From Lemma 5.6.2, we can see that the lower bound of the number of sensor nodes within a cell is $b\log n$, where $b = \arg\max_{\xi<0} \frac{ce^{\xi+1}\lambda_p - ce\lambda_p + 2}{\xi}$. Based on Lemma 5.6.1 and Lemma 5.6.2, we can obtain the capacity bounds of CAS and LAS, which are both order optimal, under the distribution model where all the nodes are deployed according to a Poisson point process as follows.

Theorem 5.6.1 *Under the Poisson point process distribution model, the achievable network capacity of CAS is $\Omega(\frac{p_o n}{\log n + 2\omega\sqrt{n/e\log n}} \cdot W) = \Omega(\frac{p_o\sqrt{en\log n}}{2\omega})$ in the best case, average case, and worst case, which is order optimal.*

Theorem 5.6.2 *Under the Poisson point process distribution model, the achievable network capacity of LAS to gather the aggregation values of N continuous snapshots is*

$$\begin{cases} \Omega(\frac{p_o N}{\vartheta}\sqrt{\frac{n}{\log n}} \cdot W), & \text{if } N = O(\sqrt{\frac{n}{\log n}}); \\ \Omega(\frac{p_o}{\vartheta\omega}\frac{n}{\log n} \cdot W), & \text{if } N = \Omega(\sqrt{\frac{n}{\log n}}). \end{cases}, \tag{5.41}$$

where

$$\vartheta = \begin{cases} 2a\omega = 2\omega\arg\min_{\xi>0} \frac{ce^{\xi+1}\lambda_p - ce\lambda_p + 2}{\xi}, & \text{in the worst case;} \\ 2ce\lambda_p\omega, & \text{in the average case;} \\ 2b\omega = 2\omega\arg\max_{\xi<0} \frac{ce^{\xi+1}\lambda_p - ce\lambda_p + 2}{\xi}, & \text{in the best case.} \end{cases} \tag{5.42}$$

and the achievable capacity is order optimal in all the cases.

5.7 Simulations

In this section, we validate the effectiveness of CAS and LAS via simulations. The simulations are conducted on a home-made simulator, which is implemented by VC++. Basically, the simulator consists of several modules involving the network generation module, the network time/synchronization control module, the network topology control module, the protocol module, etc. In all the simulations, we consider a WSN with one sink and all the sensors randomly distributed in a square area. The network time is slotted, and each time slot is normalized to one. All the nodes transmit data with a fixed power, denoted by P. Furthermore, all the sensors work on a common wireless channel with bandwidth also normalized to one. During each snapshot, every sensor node produces a packet with size one. The aggregation functions are assumed to be *perfect data aggregation functions*, i.e. the aggregation value of multiple data packets from the same snapshot is expressed using a packet of size one. Moreover, we define the node density of a WSN as ρ, i.e. on average, there are ρ sensors within a unit area. Throughout this section, we set $\rho = 5.0$ as default, which implies the WSNs with different sizes have different numbers of sensors. For other important system parameters, we set $\alpha = 3.0$, $\eta_1 = 0.25$, $\eta_2 = 10.0$, $\eta_3 = 10.0$, $P/N_0 = 10.0$, and $N = 100$. Furthermore, each group of simulations is repeated 100 times and the results are the average values.

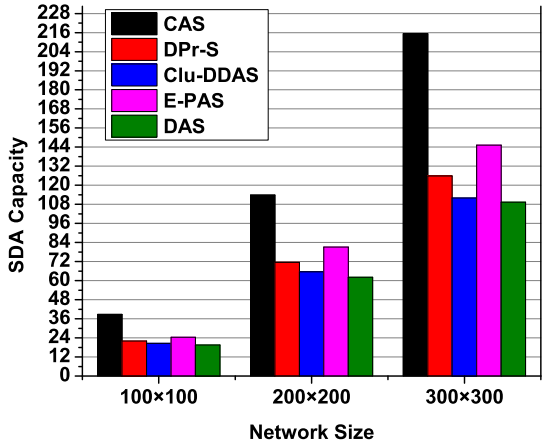
Since there are no existing works studying the SDA or CDA problems for probabilistic WSNs, we compare our proposed algorithms with the most recently published data aggregation algorithms for deterministic WSNs. Specifically, we compare our SDA algorithm CAS with DPr-S proposed in [36], Clu-DDAS proposed in [65], E-PAS proposed in [7], and DAS proposed in [63]. DPr-S is an SDA algorithm under the protocol interference model, which is a simplified interference model for ease of analysis, for deterministic WSNs [36]. Clu-DDAS is an energy-efficient algorithm for minimum-latency data aggregation scheduling under the protocol interference model for WSNs [65]. E-PAS is an SDA algorithm with the best known delay performance under the unit disk graph model, which is also analyzed under the protocol

interference model [7]. DAS is a distributed data aggregation algorithm, which is designed under the unit disk graph model and protocol interference model for WSNs [63]. For our CDA algorithm LAS, we only compare it with DPr-C [36], which is the pipelined version of DPr-S. This is because most existing works are dedicated for the SDA problem, and it is nontrivial to extend Clu-DDAS, E-PAS, and DAS to their pipelined versions. According to [36], when gathering the aggregation values of continuous snapshots, DPr-C also forms a data aggregation/transmission pipeline in three phases. In contrast, the data aggregation/-transmission pipeline in LAS is formed based on segments and scheduled based on cell-level classes and equivalent color classes, which has only one phase.

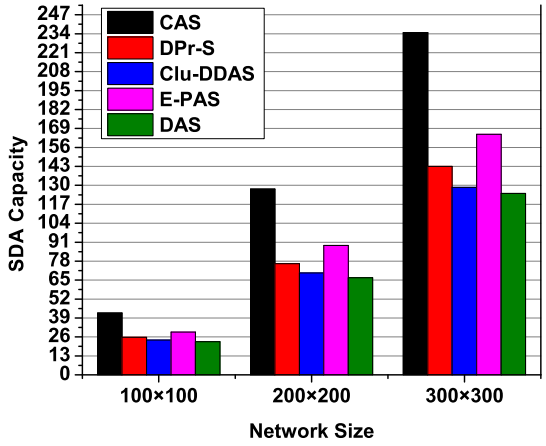
5.7.1 Performance of CAS

The achievable capacities of CAS, DPr-S, Clu-DDAS, E-PAS, and DAS in WSNs with different sizes (e.g. 100×100, 200×200, and 300×300) and different promising transmission threshold probabilities p_o are shown in Figure 5.3 and Figure 5.4. From Figure 5.3, we can see that with the increase of the network size, the achievable capacities of CAS, DPr-S, Clu-DDAS, E-PAS, and DAS also increase. This is because of the benefit brought by data aggregation. In large WSNs, more cells can conduct data aggregation operations concurrently, which implies within a time slot, more data can be aggregated. It results in increasing the data aggregation capacity. This is also validated by Theorem 5.4.1.

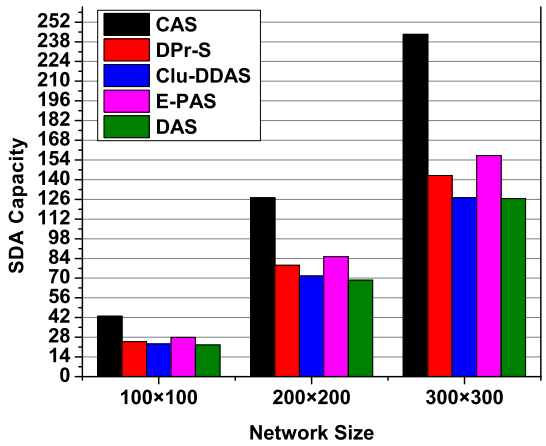
From Figure 5.4, we can also see that with the increase of p_o, the achievable capacities of CAS, DPr-S, Clu-DDAS, E-PAS, and DAS increase at first. However, after some threshold p_o, the achievable capacities of CAS, DPr-S, Clu-DDAS, E-PAS, and DAS decrease with the increase of p_o. For instance, in the WSN with size 200×200 shown in Figure 5.4(b), when p_o increases from 0.6 to 0.85, the achievable capacity of CAS increases. After that, the capacity of CAS decreases with the increase of p_o. This is because first, when p_o increases, the expected number of time slots to successfully transmit a data packet, i.e. t_n, decreases, which implies that the total number of time slots used to gather the aggregation values of a snapshot decreases. It follows that the capacities of CAS, DPr-S, Clu-DDAS, E-PAS,



(a) SDA capacity ($p_o = 0.6$).

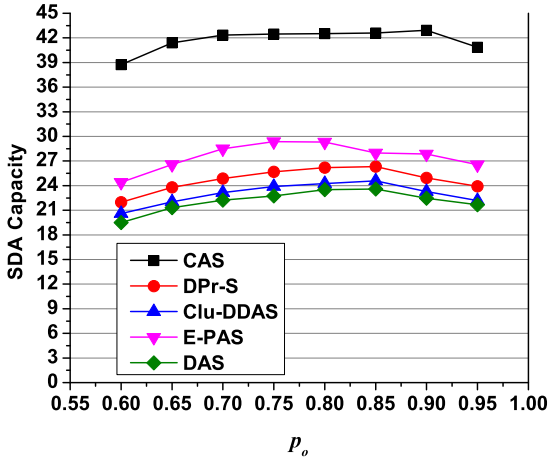


(b) SDA capacity ($p_o = 0.75$).

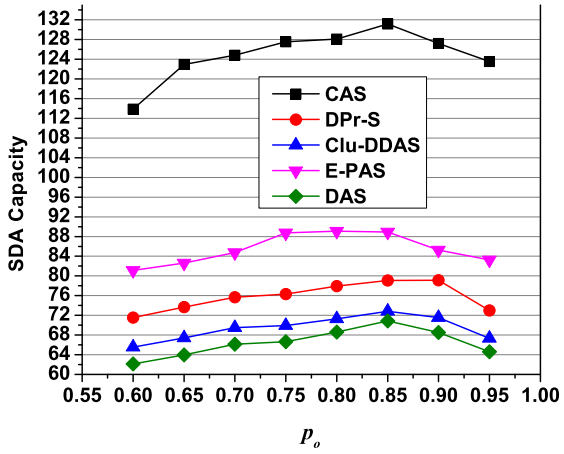


(c) SDA capacity ($p_o = 0.9$).

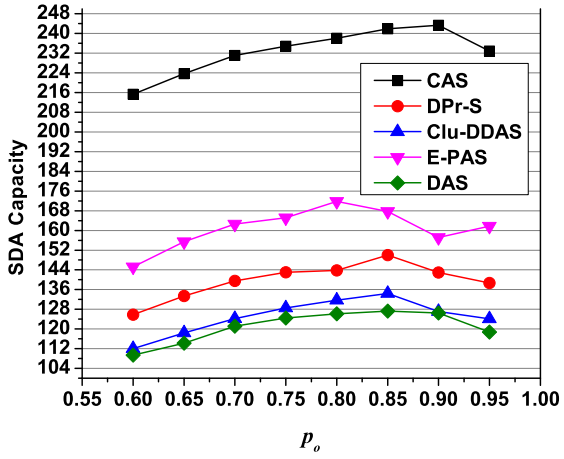
Figure 5.3 SDA capacity vs. network size (the node density $\rho = 5.0$).



(a) SDA capacity in a WSN of size 100×100.



(b) SDA capacity in a WSN of size 200×200.



(c) SDA capacity in a WSN of size 300×300.

Figure 5.4 SDA capacity vs. p_o (the node density $\rho = 5.0$).

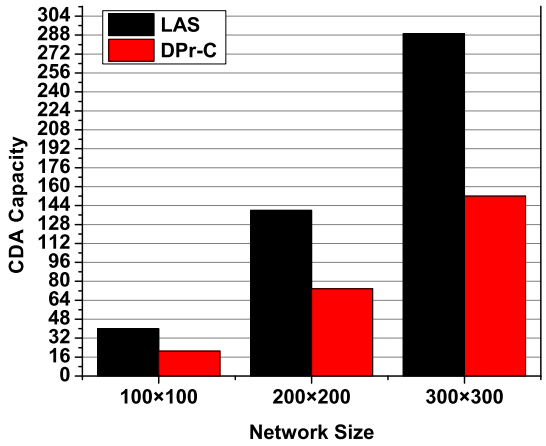
and DAS increase. Second, since a large p_o implies high-quality data communication, i.e. on the other hand, less cells can conduct aggregation operations concurrently. Thus, the network capacities of CAS, DPr-S, Clu-DDAS, E-PAS, and DAS decrease after exceeding some threshold. Note that in the WSN shown in Figure 5.4(b), for the cases of $p_o = 0.65$ and $p_o = 0.95$, although CAS achieves a similar capacity, they have quite different meanings. Since a small p_o implies more retransmission times to successfully transmit a data packet, CAS consumes more energy for the case $p_o = 0.65$ although it has similar network capacity as in the case $p_o = 0.95$.

Finally, as shown in Figure 5.3 and Figure 5.4, CAS always achieves a larger network capacity than DPr-S, Clu-DDAS, E-PAS, and DAS. This is because of the network partition methods and the equivalent color class-based scheduling scheme of CAS. By scheduling all the cells in an equivalent color class, CAS achieves complete concurrency. On the other hand, in DPr-S, Clu-DDAS, E-PAS, and DAS, either the data aggregation tree is not balanced, or the wireless channel is under-utilized, i.e. full concurrency cannot be achieved. Therefore, low SDA capacity is induced. On average, CAS achieves 67.2% more capacity than that of DPr-S, 82.46% more capacity than that of Clu-DDAS, 47.38% more capacity than that of E-PAS, and 89.65% more capacity than that of DAS.

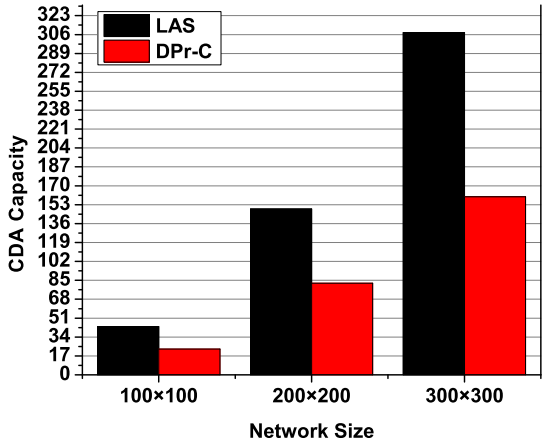
5.7.2 Performance of LAS

To gather the aggregation values of $N = 100$ continuous snapshots, the achievable capacities of LAS and DPr-C in WSNs with different sizes and promising transmission threshold probabilities are shown in Figure 5.5 and Figure 5.6, respectively. It shows in Figure 5.5 that the achievable capacities of LAS and DPr-C increase with the increase of the network size. This is because first, as mentioned before, large WSNs imply more cells can conduct aggregation operations concurrently. Second, the formed data aggregation pipelines in LAS and DPr-C perform better in large WSNs, since large WSNs can be partitioned into more segments, which are more suitable to form a pipeline to achieve higher concurrency.

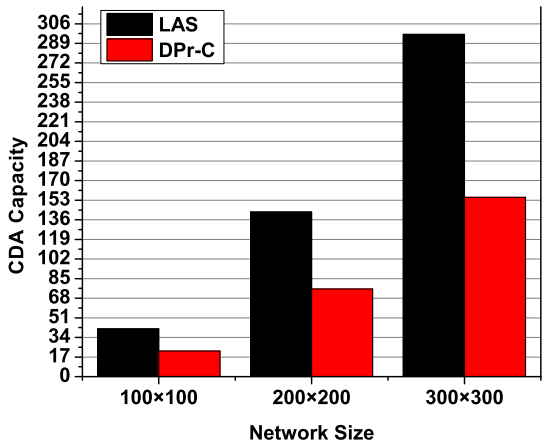
From Figure 5.6, we can also see that due to the same reasons as discussed in the previous



(a) CDA capacity ($p_o = 0.6$).

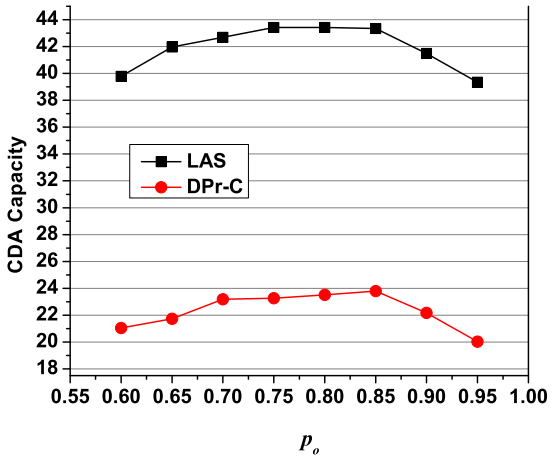


(b) CDA capacity ($p_o = 0.75$).

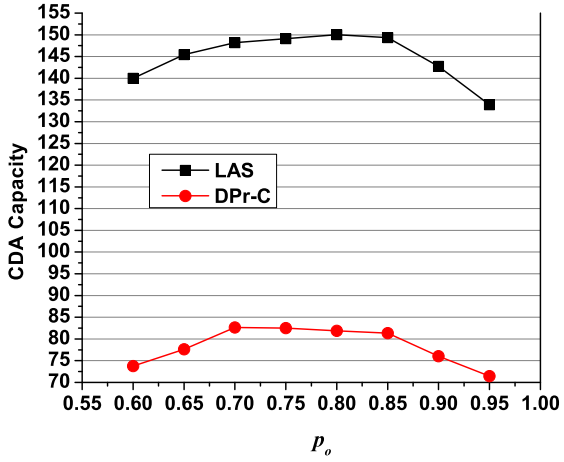


(c) CDA capacity ($p_o = 0.9$).

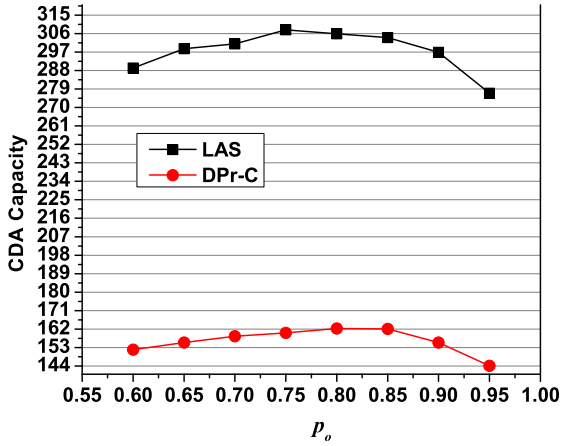
Figure 5.5 CDA capacity vs. network size (the node density $\rho = 5.0$).



(a) CDA capacity in a WSN of size 100×100.



(b) CDA capacity in a WSN of size 200×200.



(c) CDA capacity in a WSN of size 300×300.

Figure 5.6 CDA capacity vs. p_o (the node density $\rho = 5.0$).

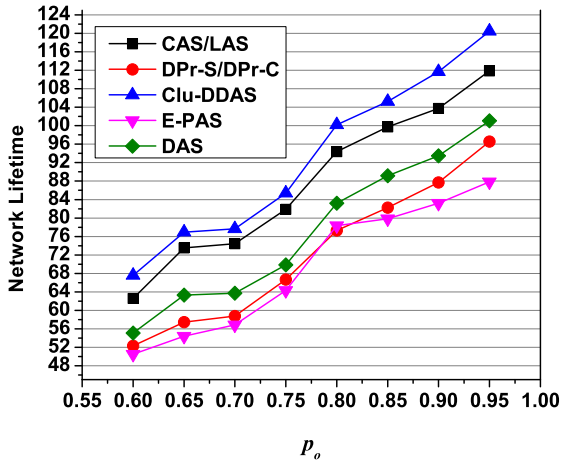
subsection, with the increase of p_o, the achievable capacities of LAS and DPr-C have similar increase and decrease trends as that of CAS and DPr-S. Furthermore, comparing Figure 5.6(a) with Figure 5.5, it is interesting to see that in some specific cases (e.g. in a WSN with size 100×100, when $p_o = 0.95$), the capacities of LAS and DPr-C are smaller than the capacities of CAS and DPr-S, respectively. This is because that the benefit of the data aggregation pipelines formed in CAS and DPr-S is not significant in small WSNs, where a pipeline is hard to form. This further validates that pipeline is more suitable for large scale WSNs.

Similar to CAS, LAS achieves complete concurrency since it schedules all the cell-levels within a cell-level class concurrently and all the cells within an equivalent color class concurrently. This turns out that LAS always achieves a higher network capacity than DPr-C as shown in Figure 5.5 and Figure 5.6. Particularly, LAS achieves 87.62% more capacity than that of DPr-C on average.

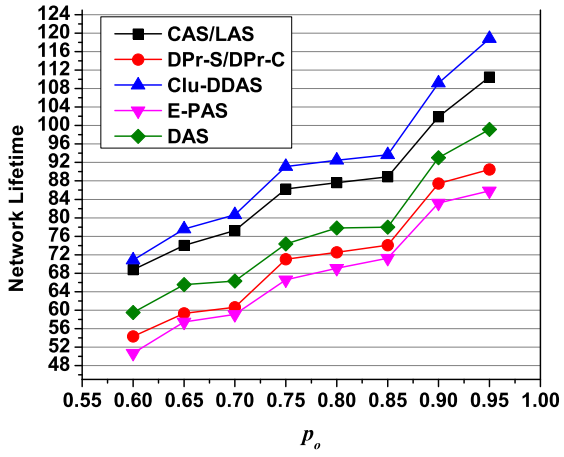
5.7.3 Network Lifetime Evaluation for CAS and LAS

In this subsection, we evaluate the network lifetime performance of data aggregation WSNs working with CAS/LAS. From the descriptions of CAS and LAS, we know that CAS and LAS have the same energy-efficiency performance, i.e. they consume the same amount of energy to gather the aggregation values of N snapshots, although LAS is much faster than CAS due to its formed data aggregation pipeline. Therefore, we consider CAS and LAS together when we evaluate their network lifetime performance. Similarly, DPr-S and DPr-C can be considered together.

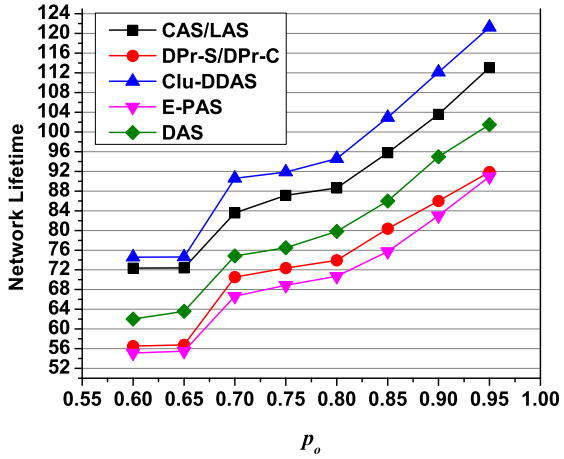
When deploy a WSN, we assume each sensor node has 1000 units of energy and the sink node has unlimited energy supply. Transmitting a data packet consumes 1 unit of energy and receiving a data packet consumes 0.5 units of energy. We further assume the energy consumption of data aggregation processing is negligible compared with that of data transmission and reception. The *network lifetime* is defined as the duration from the initial network deployment to the time when the first node exhausts its energy. We let each algo-



(a) Network lifetime of a WSN of size 100×100.



(b) Network lifetime of a WSN of size 200×200.



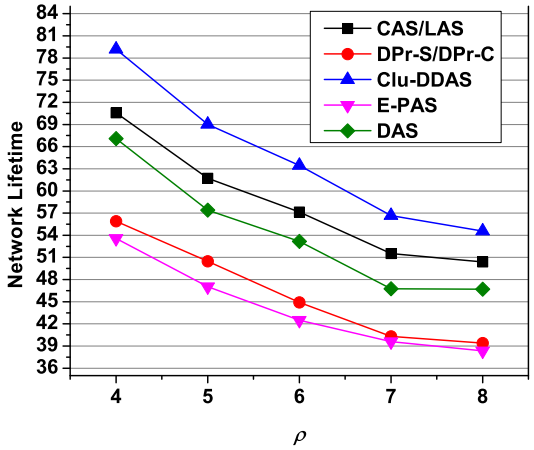
(c) Network lifetime of a WSN of size 300×300.

Figure 5.7 Network lifetime vs. p_o (the node density $\rho = 5.0$).

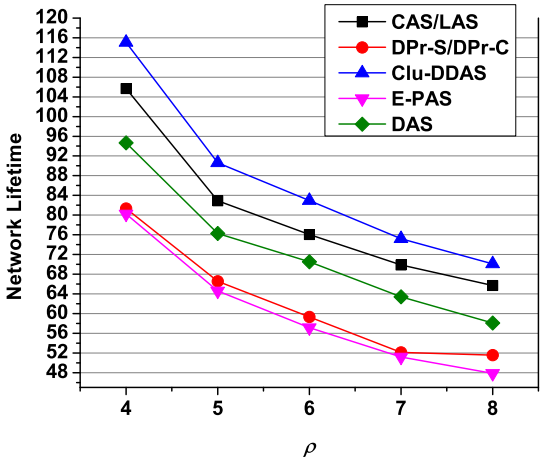
rithm continuously gather the aggregation values of snapshots until the network lifetime is ended.

Under the aforementioned assumptions, the impacts of the threshold probability p_o on the network lifetimes of CAS/LAS, DPr-S/DPr-C, Clu-DDAS, E-PAS, and DAS are shown in Figure 5.7. From Figure 5.7, we can find that the network lifetimes of all the algorithms are prolonged when p_o increases. The reason directly come from the fact that large p_o corresponding to more reliable links implies fewer number of transmission failures and retransmissions. This further implies that the energy of each node can be utilized more effectively to extend network lifetime. By comparing Figure 5.7(a), (b), and (c), we can see that network size has little impact on network lifetime. This is because we fix the node density to be $\rho = 5.0$ and all the nodes are independent and identically distributed, which implies the expected number of neighbors/children of the inner-nodes in an aggregation tree remains unchanged no matter what the size of a network is. Therefore, the traffic load of each single node in WSNs with different sizes keeps unchanged. From Figure 5.7, we can also see that Clu-DDAS has the best network lifetime performance. This is because Clu-DDAS constructs an energy-efficient cluster-based data aggregation tree. CAS/LAS achieves longer network lifetime than DPr-S/DPr-C, E-PAS, and DAS. This is because first, the data aggregation tree in CAS/LAS is balanced, while the data aggregation trees in E-PAS and DAS are imbalanced which may induce skew energy consumptions decreasing network lifetime; second, the routing structure of CAS/LAS is similar to a shortest path routing tree, while DPr-S/DPr-C first gathers data vertically and then horizontally which may induce unnecessary energy consumption.

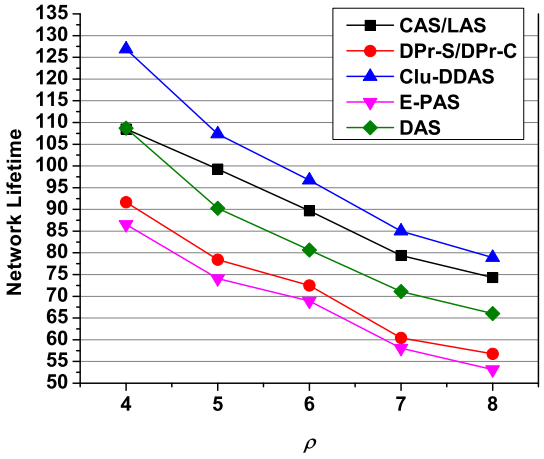
We also examine the impact of node density ρ on network lifetime for CAS/LAS, DPr-S/DPr-C, Clu-DDAS, E-PAS, and DAS as shown in Figure 5.8. From Figure 5.8, we can see that the achievable network lifetime of all the algorithms decreases when ρ increases. This is because that large ρ induces more potential workload to the inner nodes of an aggregation tree. Therefore, each inner node consumes more energy to receive local aggregation values of its children and thus the network lifetime may be decreased. From Figure 5.8, we can



(a) Network lifetime ($p_o = 0.6$).



(b) Network lifetime ($p_o = 0.75$).



(c) Network lifetime ($p_o = 0.9$).

Figure 5.8 Network lifetime vs. node density ρ (the network size is 200×200).

also see that all the algorithms perform better in WSNs with large threshold value p_o and CAS/LAS has longer network lifetime than DPr-S/DPr-C, E-PAS, and DAS. The reasons are the same as the aforementioned ones.

5.8 Conclusion

Considering that there are no existing works studying the data aggregation problem in probabilistic WSNs, we investigate the SDA and CDA problems under the PNM in this work. First, we partition a WSN into cells and equivalent color classes. Then, based on the partitioned cells and equivalent color classes, we propose a data aggregation algorithm for the SDA problem, named Cell-based Aggregation Scheduling (CAS). The theoretical analysis of CAS shows that its achievable network capacities are all $\Omega(\frac{p_o\sqrt{en\log n}}{2\omega}\cdot W)$ in the worst case, in the average case, and in the best case. Moreover, we study the upper bound capacity of the SDA problem, which is $O(\frac{p_o\sqrt{en\log n}}{3}\cdot W)$. This implies that CAS has successfully achieved order optimal capacities in all the cases. For the CDA problem, we propose a Level-based Aggregation Scheduling (LAS) algorithm. LAS achieves full concurrency by forming a data aggregation/transmission pipeline and scheduling all the cell-levels within a cell-level class simultaneously. The theoretical analysis of LAS and the CDA problem shows that LAS also successfully achieves order optimal capacities in all the cases. To be more general, we analyze the capacity performance of CAS and LAS under the non-i.i.d. node distribution model, e.g. poisson point distribution model. It shows that CAS and LAS can achieve order optimal capacities. The extensive simulation results further validate the effectiveness of CAS and LAS.

The future work may involve the following directions. First, we will extend CAS and LAS to more non-i.i.d. node distribution models and theoretically analyze their performances. Second, to obtain more accurate and tighter SDA and CDA capacity bounds, we may find some better stochastic functions to characterize the properties of lossy links. Third, since large-scale WSNs as well as other large-scale wireless networks are more likely to be distributed systems, we will design corresponding distributed and asynchronous SDA

and CDA algorithms with order optimal capacity bounds. Finally, we would like to design energy-efficient SDA and CDA algorithms which can also achieve order optimal capacities.

PART 6

CONCLUSIONS

In this dissertation, we study the data collection and aggregation problems, as well as their achievable network capacities, for WSNs.

First, we investigate the continuous data collection problem for dual-radio multi-channel WSNs under the protocol interference model. We propose a multi-path scheduling algorithm for snapshot data collection in single-radio multi-channel WSNs and derive its network capacity, which is a tighter lower bound compared with the previously best result. We subsequently propose a novel CDC method for dual-radio multi-channel WSNs. It significantly speeds up the data collection process, and achieves a capacity of $\frac{nW}{12M\lceil(3.63\rho^2+c_3\rho+c_4)/H\rceil}$ when $\Delta_e \leq 12$ or $\frac{nW}{M\Delta_e\lceil(3.63\rho^2+c_3\rho+c_4)/H\rceil}$ when $\Delta_e > 12$, where n is the number of the sensors, M is a constant value and usually $M \ll n$, Δ_e is the maximum number of the leaf nodes having a same parent in the data collection tree, W is the channel bandwidth, H is the number of available orthogonal channels, ρ is the ratio of the interference radius over the transmission radius, $c_3 = \frac{8\pi}{\sqrt{3}} + \pi + 2$, and $c_4 = \frac{8\pi}{\sqrt{3}} + 2\pi + 6$. Extensive simulation results indicate that the proposed algorithms improve network capacity significantly compared with existing works.

Second, considering that for most existing works studying the network capacity issue, their designs and analysis are based on the deterministic network model, where any pair of nodes in a network is either "*connected*" or "*disconnected*". However, in real application environments, this deterministic network model assumption is too ideal and not practical due to the existence of the "*transitional region phenomenon*". Actually, a more practical network model for wireless networks is the *probabilistic network model*, where a transmission over a link is conducted successfully with a probability instead of being determined. Unfortunately, few of the existing works study the data collection capacity issue for wireless networks under the practical probabilistic network model until now. To remedy this gap, we investigate

the achievable snapshot/continuous data collection capacity for wireless networks under the probabilistic network model. For snapshot data collection, we propose a novel *Cell-based Path Scheduling* (CPS) algorithm which achieves capacity of $\Omega(\frac{1}{5\omega \ln n} \cdot W)$ in the sense of the worst case and order-optimal capacity in the sense of expectation, where n is the number of sensor nodes, ω is a constant, and W is the data transmitting rate. For continuous data collection, we propose a *Zone-based Pipeline Scheduling* (ZPS) algorithm. ZPS significantly speeds up the continuous data collection process by forming a data transmission pipeline, and achieves a capacity gain of $\frac{N\sqrt{n}}{\sqrt{\log n} \ln n}$ or $\frac{n}{\log n \ln n}$ times better than the optimal capacity of the snapshot data collection scenario in order in the sense of the worst case, where N is the number of snapshots in a continuous data collection task. The simulation results also validate that the proposed algorithms significantly improve network capacity compared with the existing works.

Third, most of the existing works studying the data collection capacity issue have an ideal assumption that the network time is synchronized explicitly or implicitly. Such an assumption is mainly for centralized synchronous wireless networks. However, wireless networks are more likely to be distributed asynchronous systems. Thus, we investigate the achievable data collection capacity of realistic *distributed asynchronous* WSNs. Our main contributions include five aspects. Firstly, to avoid data transmission interference, we derive an $\mathcal{R}_0$-*Proper Carrier-sensing Range* ($\mathcal{R}_0$-PCR) under the *generalized physical interference model*, where $\mathcal{R}_0$ is the *satisfied threshold of data receiving rate*. Taking $\mathcal{R}_0$-PCR as its *carrier-sensing range*, any sensor node can initiate a data transmission with a guaranteed data receiving rate. Secondly, based on $\mathcal{R}_0$-PCR, we propose a *Distributed Data Collection* (DDC) algorithm with fairness consideration. Theoretical analysis of DDC surprisingly shows that its achievable network capacity is order-optimal and independent of network size. Thus, DDC is scalable. Thirdly, we discuss how to apply $\mathcal{R}_0$-PCR to the distributed data aggregation problem, and propose a *Distributed Data Aggregation* (DDA) algorithm. The delay performance of DDA is also analyzed. Fourthly, to be more general, we study the delay and capacity of DDC and DDA under the Poisson node distribution model. The analysis

demonstrates that DDC is also scalable and order-optimal under the Poisson distribution model. Finally, we conduct extensive simulations to validate the performance of DDC and DDA.

Fourth, we study the *Snapshot Data Aggregation* (SDA) problem, the *Continuous Data Aggregation* (CDA) problem, and their achievable capacities for probabilistic WSNs under both the independent and identically node distribution (i.i.d.) model and the Poisson point distribution model in this dissertation. First, we partition a network into *cells* and use two vectors to further partition these cells into *equivalent color classes*. Subsequently, based on the partitioned cells and equivalent color classes, we propose a *Cell-based Aggregation Scheduling* (CAS) algorithm for the SDA problem in probabilistic WSNs. Theoretical analysis of CAS and the upper bound capacity of the SDA problem show that the achievable capacities of CAS are all order optimal in the worst case, the average case, and the best case. For the CDA problem in probabilistic WSNs, we propose a *Level-based Aggregation Scheduling* (LAS) algorithm. LAS gathers the aggregation values of continuous snapshots by forming a data aggregation/transmission pipeline on the *segments* and scheduling all the *cell-levels* in a *cell-level class* concurrently. By theoretical analysis of LAS and the upper bound capacity of the CDA problem, we prove that LAS also successfully achieves order optimal capacities in all the cases. The extensive simulation results further validate the effectiveness of CAS and LAS. Specifically, compared with the most recently published algorithm, CAS achieves 67.95% more capacity than that of DPr-S on average, and LAS achieves 90.45% more capacity than that of DPr-C on average.

REFERENCES

[1] J. S. C. Luo, F. Wu and C. W. Chen, "Compressive data gathering for large-scale wireless sensor networks," *Proceedings of the Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2009.

[2] Y. S. I. F. Akyildiz, W. Su and E. Cayirci, "Wireless sensor networks: A survey," *Proceedings of the Elsevier Computer Networks*, vol. 38, 2002.

[3] "http://en.wikipedia.org/wiki/wireless_sensor_network."

[4] M. H. S. Chen, S. Tang and Y. Wang, "Capacity of data collection in arbitrary wireless sensor networks," *Proceedings of the Annual IEEE International Conference on Computer Communications (INFOCOM)*, 2010.

[5] Y. L. S. Ji and X. Jia, "Capacity of dual-radio multi-channel wireless sensor networks for continuous data collection," *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2011.

[6] J. L. S. Cheng and Z. Cai, "$o(\varepsilon)$-approximation to physical world by sensor networks," *Proceedings of the Annual IEEE International Conference on Computer Communications (INFOCOM)*, 2013.

[7] L. W. Z. W. P.-J. Wan, S. C.-H. Huang and X. Jia, "Minimum-latency aggregation scheduling in multihop wireless networks," *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, 2009.

[8] Y. P. S. Ji, J. He and Y. Li, "Continuous data aggregation and capacity in probabilistic wireless sensor netwroks," *Proceedings of the Journal of Parallel and Distributed Computing (JPDC)*, vol. 73, no. 6, pp. 729–745, 2013.

[9] C. Liu and G. Cao, "Distributed monitoring and aggregation in wireless sensor networks," *Proceedings of the Annual IEEE International Conference on Computer Communications (INFOCOM)*, 2010.

[10] S. J. M. Yan, J. He and Y. Li, "Minimum latency scheduling for multi-regional query in wireless sensor networks," *Proceedings of the IEEE International Performance Computing and Communications Conference (IPCCC)*, 2011.

[11] ——, "Multi-regional query scheduling in wireless sensor networks with minimum latency," *Proceedings of the Wireless Communications and Mobile Computing (WCMC)*, 2012.

[12] S. T. X.-Y. Li and O. Frieder, "Multicast capacity for large scale wireless ad hoc networks," *Proceedings of the Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2007.

[13] Y. L. S. Li and X.-Y. Li, "Capacity of large scale wireless networks under gaussian channel model," *Proceedings of the Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2008.

[14] X.-Y. L. X. Mao and S. Tang, "Multicast capacity for hybrid wireless networks," *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, 2008.

[15] P. G. U. Niesen and D. Shah, "On capacity scaling in arbitrary wireless networks," *Proceedings of the IEEE Transactions on Information Theory*, vol. 55, no. 9, pp. 3959–3982, 2009.

[16] ——, "The balanced unicast and multicast capacity regions of large wireless networks," *Proceedings of the IEEE Transactions on Information Theory*, vol. 56, no. 5, pp. 2249–2271, 2010.

[17] M. C. C.-K. Chau and S. C. Liew, "Capacity of large-scale csma wireless networks," *Proceedings of the Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2009.

[18] Y. W. W. S. T. X. H. X. X.-Y. Li, J. Zhao and X. F. Mao, "Broadcast capacity for wireless ad hoc networks," *Proceedings of the IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, 2008.

[19] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *Proceedings of the IEEE Transactions on Information Theory*, vol. 46, no. 2, pp. 388–404, 2000.

[20] V. S. A. K. M. V. M. S. P. D. Chafekar, D. Levin and A. Srinivasan, "apacity of asynchronous random-access scheduling in wireless networks," *Proceedings of the Annual IEEE International Conference on Computer Communications (INFOCOM)*, 2008.

[21] M. Andrews and M. Dinitz, "Maximizing capacity in arbitrary wireless networks in the sinr model: Complexity and game theory," *Proceedings of the Annual IEEE International Conference on Computer Communications (INFOCOM)*, 2009.

[22] M. M. H. O. Goussevskaia, R. Wattenhofer and E. Welzl, "Capacity of arbitrary wireless networks," *Proceedings of the Annual IEEE International Conference on Computer Communications (INFOCOM)*, 2009.

[23] H. R. S. Z. Wang and J. J. Garcia-Luna-Aceves, "A unifying perspective on the capacity of wireless ad hoc networks," *Proceedings of the Annual IEEE International Conference on Computer Communications (INFOCOM)*, 2008.

[24] X. W. W. Huang and Q. Zhang, "Capacity scaling in mobile wireless ad hoc network with infrastructure support," *Proceedings of the International Conference on Distributed Computing Systems (ICDCS)*, 2010.

[25] X. W. G. Zhang, Y. Xu and M. Guizani, "apacity of hybrid wireless networks with

directional antenna and delay constraint," *Proceedings of the IEEE Transactions on Communications*, vol. 58, no. 7, pp. 2097–2016, 2010.

[26] S. P. V. S. A. Kumar, M. V. Marathe and A. Srinivasan, "Algorithmic aspects of capacity in wireless networks," *Proceedings of the ACM Special Interest Group on Measurement and Evaluation (SIGMETRICS)*, 2005.

[27] V. R. A. Keshavarz-Haddad and R. Riedi, "Broadcast capacity in multihop wireless networks," *Proceedings of the Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2006.

[28] D. T. B. Liu and A. Swami, "Data gathering capacity of large scale multihop wireless networks," *Proceedings of the IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, 2008.

[29] C. L. O. Chipara and J. Stankovic, "Dynamic conflict-free query scheduling for wireless sensor networks," *Proceedings of the IEEE International Conference on Network Protocols (ICNP)*, 2006.

[30] E. J. Duarte-Melo and M. Liu, "Data-gathering wireless sensor networks: Organization and capacity," *Proceedings of the Computer Networks*, vol. 43, 2003.

[31] M. L. D. Marco, E. J. Duarte-Melo and D. L. Neuhoff, "On the many-to-one transport capacity of a dense wireless sensor network and the compressibility of its data," *Proceedings of the ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2003.

[32] H. E. Gamal, "On the scaling laws of dense wireless sensor networks," *Proceedings of the IEEE Transactions on Information Theory*, vol. 51, no. 3, pp. 1229–1234, 2003.

[33] Q. P. X. Wang, Y. Bei and L. Fu, "peed improves delay-capacity tradeoff in motioncast," *Proceedings of the IEEE Transactions on Parallel and Didstributed Systems*, 2011.

[34] D. N. C. T. M. Franceschetti, O. Dousse and P. Thiran, "Closing the gap in the capacity of wireless networks via percolation theory," *Proceedings of the IEEE Transactions on Information Theory*, vol. 53, no. 3, pp. 1009–1018, 2007.

[35] B. T. X. Zhu and H. Gupta, "Delay efficient data gathering in sensor networks," *Proceedings of the International Conference on Mobile Ad-hoc and Sensor Networks (MSN)*, 2005.

[36] X.-Y. L. S. Chen, Y. Wang and X. Shi, "Order-optimal data collection in wireless sensor networks: Delay and capacity," *Proceedings of the Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, 2009.

[37] ——, "Data collection capacity of random-deployed wireless sensor networks," *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*, 2009.

[38] R. B. S. Ji and Y. Li, "Continuous data collection capacity of wireless sensor networks under physical interference model," *Proceedings of the IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, 2011.

[39] R. B. S. Ji and Z. Cai, "Snapshot/continuous data collection capacity for large-scale probabilistic wireless sensor networks," *Proceedings of the Annual IEEE International Conference on Computer Communications (INFOCOM)*, 2012.

[40] ——, "Snapshot and continuous data collection in probabilistic wireless sensor networks," *Proceedings of the IEEE Transactions on Mobile Computing (TMC)*, 2013.

[41] T. Moscibroda, "The worst-case capacity of wireless sensor networks," *Proceedings of the ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2007.

[42] S. Ji and Z. Cai, "Distributed data collection and its capacity in asynchronous wire-

less sensor networks," *Proceedings of the Annual IEEE International Conference on Computer Communications (INFOCOM)*, 2012.

[43] R. B. S. Ji, A. S. Uluagac and Z. Cai, "Practical unicast and convergecast scheduling schemes for cognitive radio networks," *Proceedings of the Journal of Combinatorial Optimization (JCO)*, 2012.

[44] J. H. Z. Cai, S. Ji and A. G. Bourgeois, "Optimal distributed data collection for asynchronous cognitive radio networks," *Proceedings of the International Conference on Distributed Computing Systems (ICDCS)*, 2012.

[45] J. H. L. W. Z. Cai, S. Ji and A. G. Bourgeois, "Distributed and asynchronous data collection in cognitive radio networks with fairness consideration," *Proceedings of the IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 2013.

[46] H. R. S. Z. Wang and J. J. Garcia-Luna-Aceves, "The capacity and energy efficiency of wireless ad hoc networks with multi-packet reception," *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, 2008.

[47] Y. Xu and W. Wang, "Scheduling partition for order optimal capacity in large-scale wireless networks," *Proceedings of the Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2009.

[48] P. G. M. Garetto and E. Leonardi, "On the capacity of ad hoc wireless networks under general node mobility," *Proceedings of the Annual IEEE International Conference on Computer Communications (INFOCOM)*, 2007.

[49] R. M. G. Sharma and N. B. Shroff, "Delay and capacity trade-offs in mobile ad hoc networks: A global perspective," *Proceedings of the IEEE/ACM Transactions on Networking*, vol. 15, no. 5, pp. 981–992, 2007.

[50] V. S. A. K. A. Ghosh, Ö. D. Incel and B. Krishnamachari, "Multi-channel scheduling algorithms for fast aggregated covergecast in sensor networks," *Proceedings of the IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, 2009.

[51] R. B. M. Alicherry and L. E. Li, "Joint channel assignment and routing for throughput optimization in multi-radio wireless mesh networks," *Proceedings of the Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2005.

[52] M. V. M. S. P. B. Han, V. S. Anil Kumar and A. Srinivasan, "Distributed strategies for channel allocation and scheduling in software-defined radio networks," *Proceedings of the Annual IEEE International Conference on Computer Communications (INFOCOM)*, 2009.

[53] X. Lin and S. B. Rasool, "Distributed and provably efficient algorithms for joint channel-assignment, scheduling, and routing in multichannel ad hoc wireless networks," *Proceedings of the IEEE/ACM Transcations on Networking*, vol. 17, no. 6, pp. 1874–1886, 2009.

[54] V. Bhandari and N. H. Vaidya, "Connectivity and capacity of multi-channel wireless networks with channel switching constraints," *Proceedings of the Annual IEEE International Conference on Computer Communications (INFOCOM)*, 2007.

[55] ——, "Capacity of multi-channel wireless networks with random (c, f) assignment," *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, 2007.

[56] A. C. V. Ramamurthi, S. K. C. Vadrevu and M. R. Bhatnagar, "Multicast capacity of multi-channel multihop wireless networks," *Proceedings of the WCNC*, 2009.

[57] R. C.-W. W. H.-N. Dai, K.-W. Ng and M.-Y. Wu, "On the capacity of multi-channel wireless networks using directional antennas," *Proceedings of the Annual IEEE International Conference on Computer Communications (INFOCOM)*, 2008.

[58] P. Kyasanur and N. H. Vaidya, "Capacity of multi-channel wireless networks: Impact of number of channels and interfaces," *Proceedings of the Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2005.

[59] Y. L. S. Ji, Z. Cai and X. Jia, "Continuous data collection capacity of dual-radio multi-channel wireless sensor networks," *Proceedings of the IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 23, no. 10, pp. 1844–1855, 2012.

[60] X. H. X. Chen and J. Zhu, "Minimum data aggregation time problem in wireless sensor networks," *Proceedings of the International Conference on Mobile Ad-hoc and Sensor Networks (MSN)*, 2005.

[61] C. T. V. Y. L. S. C.-H. Huang, P.-J. Wan and F. Yao, "Nearly constant approximation for data aggregation scheduling in wireless sensor networks," *Proceedings of the Annual IEEE International Conference on Computer Communications (INFOCOM)*, 2007.

[62] P.-J. W. X. H. Xu, X.-Y. Li and S. J. Tang, "Efficient scheduling for periodic aggregation queries in multihop sensor networks," *Proceedings of the IEEE/ACM Transactions on Networking*, 2011.

[63] J. L. B. Yu and Y. Li, "Distributed data aggregation scheduling in wireless sensor networks," *Proceedings of the Annual IEEE International Conference on Computer Communications (INFOCOM)*, 2009.

[64] X. F. M. S. J. T. X. H. Xu, S. G. Wang and X. Y. Li, "An improved approximation algorithm for data aggregation in multi-hop wireless sensor networks," *Proceedings of the ACM International Workshop on Foundations of Wireless Ad Hoc and Sensor Networking and Computing (FOWANC)*, 2009.

[65] L. G. Y. Li and S. K. Prasad, "An energy-efficient distributed algorithm for minimum-latency aggregation scheduling in wireless sensor networks," *Proceedings of the International Conference on Distributed Computing Systems (ICDCS)*, 2010.

[66] Y. D. J. H. R. B. S. Ji, M. Yan and Z. Cai, "Broadcast scheduling for cognitive radio networks," *Proceedings of the Annual IEEE International Conference on Computer Communications (INFOCOM)*, 2013.

[67] M. G. D. Lymberopoulos, N. B. Priyantha and F. Zhao, "Towards energy efficient design of multi-radio platforms for wireless sensor networks," *Proceedings of the ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2008.

[68] T. Z. X. L. W. Cheng, X. Chen and Z. Lu, "The complexity of channel scheduling in multi-radio multi-channel wireless networks," *Proceedings of the Annual IEEE International Conference on Computer Communications (INFOCOM)*, 2009.

[69] F. W. Y. Li, M. T. Thai and D.-Z. Du, "On the construction of a strongly connected broadcast arborescence with bounded transmission delay," *Proceedings of the IEEE Transactions on Mobile Computing (TMC)*, vol. 5, no. 10, pp. 1460–1470, 2006.

[70] P. F. Y. P. J. He, S. Ji and Y. Li, "Constructing a load-balanced virtual backbone in wireless sensor networks," *Proceedings of the International Conference on Computing, Networking and Communications (ICNC)*, 2012.

[71] M. Y. Y. P. J. He, S. Ji and Y. Li, "Load-balanced cds construction in wireless sensor networks via genetic algorithm," *Proceedings of the International Journal of Sensor Networks (IJSNet)*, vol. 11, no. 3, pp. 166–178, 2012.

[72] Y. P. J. He, S. Ji and Y. Li, "Greedy construction of load-balanced virtual backbones in wireless sensor networks," *Proceedings of the Wireless Communications and Mobile Computing (WCMC)*, 2012.

[73] D. W. Matula and L. L. Beck, "Smallest-last ordering and clustering and graph coloring algorithms," *Proceedings of the Journal of the Association of Computing Machinery*, vol. 30, no. 3, pp. 417–427, 1983.

[74] Q. Z. Y. Liu and L. M. Ni, "Opportunity-based topology control in wireless sensor networks," *Proceedings of the IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 3, pp. 405–416, 2010.

[75] M. Zuniga and B. Krishnamachari, "Analyzing the transitional region in low power wireless links," *Proceedings of the Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, 2004.

[76] S. Ji and Z. Cai, "Distributed data collection in large-scale asynchronous wireless sensor networks under the generalized physical interference model," *Proceedings of the IEEE/ACM Transactions on Networking (ToN)*, 2012.

[77] A. S. U. R. B. S. Ji, J. He and Y. Li, "Cell-based snapshot and continuous data collection in wireless sensor networks," *Proceedings of the ACM Transactions on Sensor Networks (TOSN)*, 2012.

[78] S. R. Kulkarni and P. Viswanath, "A deterministic approach to throughput scaling in wireless networks," *Proceedings of the IEEE Transactions on Information Theorgy*, vol. 50, no. 6, pp. 73–84, 2004.

[79] H.-C. W. S. C.-H. Huang, S. Y. Chang and P.-J. Wan, "Analysis and design of a novel randomized broadcast algorithm for scalable wireless networks in the interference channels," *Proceedings of the IEEE Transactions on Wireless Communications*, vol. 9, no. 7, pp. 2206–2215, 2010.

[80] S. C. L. L. Fu and J. Huang, "Effective carrier sensing in csma networks under cumulative interference," *Proceedings of the Annual IEEE International Conference on Computer Communications (INFOCOM)*, 2010.

[81] L. T. F. Ferrari, M. Zimmerling and O. Saukh, "Efficient network flooding and time synchronization with glossy," *Proceedings of the ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2011.